NPS ARCHIVE
1997.05
NIMITBUNANAN, N.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

VIDEO CONFERENCING
USING
PACKET RADIO TECHNOLOGY

by

Narongchai Nimitbunanan

June 1997

Thesis Advisor: Chin-Hwa Lee
Second Reader: Supachai Sirayanone

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1997	Master's Thesis

4. TITLE AND SUBTITLE
VIDEO CONFERENCING USING PACKET RADIO TECHNOLOGY

5. FUNDING NUMBERS

6. AUTHOR(S)
Nimitbunanan, Narongchai

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Information and its effective delivering means are becoming more and more important in today's world. Video-conferencing is a highly effective means to deliver information since it is interactive. This thesis studies the packet-radio-networking technology that can be used to support video-conferencing applications. The popular networking protocols, i.e. the Amateur X.25 (AX.25), the Transport Control Protocol/ Internet Protocol (TCP/IP), and other protocols, widely used in packet radio technology are described. By using the File Transfer Protocol (FTP) of the TCP/IP standard, the average speed and time of various file sizes across a half-duplex radio channel, a full-duplex emulated-radio channel, and a RS-232 link were collected and analyzed. Finally, comparisons were made among channels, including the effects of an additional routing node.

SUBJECT TERMS	15. NUMBER OF PAGES
Video Conferencing, Packet Radio, Amateur Packet Radio, AX.25, TCP/IP, TNC, Routing	102
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited

VIDEO CONFERENCING USING PACKET RADIO TECHNOLOGY

Narongchai Nimitbunanan
Second Lieutenant, Royal Thai Air Force
B.S., U.S. Air Force Academy, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1997

NPS ARCHIVE
1997.06
NIMITZBUNANAN,N.

N
0513
N5916
c.

ABSTRACT

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

Information and its effective delivering means are becoming more and more important in today's world. Video-conferencing is a highly effective means to deliver information since it is interactive. This thesis studies the packet-radio-networking technology that can be used to support video-conferencing applications. The popular networking protocols, i.e. the Amateur X.25 (AX.25), the Transport Control Protocol/ Internet Protocol (TCP/IP), and other protocols, widely used in packet radio technology are described. By using the File Transfer Protocol (FTP) of the TCP/IP standard, the average speed and time of various file sizes across a half-duplex radio channel, a full-duplex emulated-radio channel, and a RS-232 link were collected and analyzed. Finally, comparisons were made among channels, including the effects of an additional routing node.

TABLE OF CONTENTS

PAGES

I. INTRODUCTION....1

A. PROJECT OVERVIEW....1

B. INTRODUCTION....2

C. GOALS....4

D. APPROACH....5

II. BACKGROUND....7

A. PACKET RADIO BACKGROUND....7

1. History of Packet Radio....7

2. The AX.25 Protocol....9

B. PACKET RADIO NETWORKING....18

1. Network Operating System (NOS)....18

2. Networking Schemes....19

a. NET/ROM....20

b. ROSE....21

c. TCP/IP....21

III. EXPERIMENTAL SET-UPS....24

A. THE TNOS SOFTWARE....24

1. An Introduction to TNOS....24

2. Installing the TNOS....25

3. I/O Device Interfacing....27

4. Setting Up the FTP Session....28

5. Setting the TNC Parameters....28

B. THE HARDWARE....30

1. The Terminal Node Controller(TNC) & The D4-10 Transceiver....30

2. The PC and its FIFO....33

3. Hardware Interfaces & Low Level Protocols....35

IV. RESULTS....37

A. DATA PACKETS....37

B. EXPERIMENT # 1 : HALF-DUPLEX RADIO CHANNEL....41

C. EXPERIMENT # 2 : HARD-WIRED FULL DUPLEX....48

D. EXPERIMENT # 3 : ENHANCED HARD-WIRED FULL DUPLEX.... 55

E. EXPERIMENT # 4 : ROUTING EFFECTS....61

V. DISCUSSION....67

A. TIME COMPARISON....67

B. SPEED COMPARISON....69

VI. CONCLUSION AND FUTURE WORKS....75

APPENDIX A: CONFIGURATION FILES....77

APPENDIX B: COLLECTED DATA POINTS....85

LIST OF REFERENCES....91

INITIAL DISTRIBUTION LIST....93

I. INTRODUCTION

A. PROJECT OVERVIEW

Information is becoming more and more important in the modern days' day-to-day activities, whether they be civilian activities or military activities. The means to deliver information has evolved rapidly over the past years. It is true that information superiority is becoming another important pillar of any combat type of operations. Hence, there is a need for better means of delivering vital information effectively, reliably, and on-time, because these can result in life or death in combat situations. Traditionally, information is mostly delivered over voice/audio type of systems, in real-time or near real-time. However, 'a picture is worth a thousand words' as an old Chinese saying, hence, it may be beneficial to find a better mean to deliver essential information both in traditional voice/audio and in life-interactive-video, in a real-time, or near real-time mode.

In a shipboard situation, such as in a battle group where there are many ships traveling together in an open sea. The live-interactive-video conferencing may be very useful in communicating with each other among various ships. Combat effectiveness and/or logistic support activities may be improved by implementing this type of system as a mean to deliver effective, reliable, on-time, and secure information among ships in the same battle group or to a different group nearby. The system studied here is the

packet radio network, which can be used as a backbone or as a medium to deliver the information among ships. The system will have a number of nodes or routers installed on each ship. And, the information would be sent out by a source, which may be one of many ships in the battle group, and then would be routed through these nodes/routers installed on the ships, from ship-to-ship until it reaches the desired destination. However, the implementations of this system is not only restricted to a sea-based platforms but can also be implemented on lands where many nodes/routers are installed on the vehicles, which may be either moving or stationary.

B. INTRODUCTION

Since our objective is to conduct video conferencing over a packet radio network in an economic way, there are two important issues we need to consider. First is what type of hardware/software we need to capture, encode/decode, and display acceptable video-quality images. And, second is what type of networking supports we need as a medium to deliver the data. A practical and economic way to deal with the first issue is to utilize the Desk-Top-Video-Conferencing (DTVC) technology due to its low-cost and availability in today's market. A good source for DTVC software/hardware vendors is listed by C.E. Hendricks and J.P. Steer [17]. There are many available commercial hardware/software packages in today's market such as Cu-SeeMe, Picture Tel, Connectrix, VideoLabs, and etc. But, to deal with the second issue, which is the focus in our study, we need to find out what type of radio networks and supports needed to accomplish our objective.

For flexibility and compatibility with the industrial standards and today's markets, our system is intended to support the Transport Control Protocol/Internet Protocol (TCP/IP) in many networking environments such as the Wide Area Network (WAN), Local Area Network (LAN), the Internet, and the Packet Radio Networks, etc. The implementations of our system will probably be in a packet-switching type of networks. In 1994, J. Harju, V.P. Kosonen, and C. Li studied the quality and performance of DTVC technology in the interconnected LANs environments with TCP/IP carriers [18]. The team found out that there is a slight degradation in video quality with increasing number of nodes and/or switches. However, our study only focus on the effects of one node/router positioned between links with the TCP/IP protocol riding over another protocol i.e. the Amateur X.25 protocol (AX.25). The AX.25 protocol is widely used in the Amateur Packet Radio Networks and has many good features to support delivery of data over a radio-networking scheme. Also, during 1995, C.P. Bandy, D.B. Koch did some experiments on transmitting stilled images over a low-bandwidth transmission system [9]. Their system emulated a packet-radio transmission by using telephone modems. The system demonstrated promising results for delivering quality stilled images over a low-bandwidth channel. In our experiments, however, we actually transmit data packets over a real half-duplex packet-radio channel, and also over a emulated full-duplex radio channel.

There are a couple of reasons which make this project very interesting; the flexibility of the system and the possibility of a practical video-conferencing system at a

very low-cost. This project will require little or no development cost at all, since all the technologies, both hardware and software components, are already available at low costs in today's market. The intended system would use mostly Commercial-off-the-Shelf products with little or no software modifications to suit the needs.

C. GOALS

The overall goal of this project is to study a prototype system which may be used to implement an interactive video-conferencing application by using a packet radio networking scheme. The long-term goal is to incorporate commercially available video-conferencing software/hardware to our packet-radio networking backbone. The video-conferencing software/hardware components should support standard protocols such as Serial-Line-Interface Protocol (SLIP), TCP /IP, Point-to-Point Protocol (PPP), and etc. Hence, we should be able to incorporate these components in some kind of a Network-Operating-System (NOS) software which will be used to run on the nodes/router.

However, the scope of the study in this paper is only to investigate the capabilities, possibilities , and limitations of various components and transmission mediums required to support such a system. Hence, the initial study explained in this thesis is limited to only study the behaviors of the data being transmitted through the amateur radio channel (i.e. the 440-MHz UHF band). The amateur packet radio utilizes the AX.25 protocol, and the existing TCP/IP networks implemented through the directly connected RS-232 line. In addition, routing behavior of the packets through both

channels are studied because the overall goal of this continuing project is to conduct video-conferencing over the hybrid networks i.e. the packet-radio networks and the existing conventional TCP/IP networks (or other wired networks).

D. APPROACH

The approach used in this study is to study the channel's data transmission behavior by sending some data using a packet-radio protocol between various nodes connected by various types of mediums i.e. the RS-232 line, the radio half-duplex channel, and the radio full-duplex channel. First, the data packets are sent through the UHF radio channel with a maximum speed of 19.2 K-baud. Then, the channel is improved by emulating it as a full-duplex radio channel. With an emulated channel, to study the routing effects, the data packets are routed through both the link, a RS-232 line connected between the two nodes running under a Network Operating System (NOS), and the emulated-radio full-duplex channel. The application protocol used in this experiments is the File-Transfer-Protocol (FTP). Various file sizes are created and sent through the channels. Then, the performance parameters, i.e. the time in seconds and the average speed in Bytes/second for each file transfer operation, are recorded. All the networking nodes in our experiments are running under the Tampa Network Operating System (TNOS), which is just a version the original KA9Q Network Operating System (KA9Q NOS), written by Phil Karn [6].

This thesis is organized into six major chapters. The first chapter, Chapter I, describes the project in a big picture, including approach used and objectives of our studies. Chapter II gives the readers some backgrounds needed to understand fundamental technologies in this field; the packet radio and networking. Chapter III explains the experimental set ups, used in our study. Then, the results are explained in Chapter IV. Chapter V discusses the results and compare them with other experiments. Lastly, Chapter VI concludes the experiments and recommends future research efforts.

II. BACKGROUND

A. PACKET RADIO BACKGROUND

Amateur Packet Radio is an on-going development technology in the amateur radio world. Both the software and hardware are currently being developed and tested by the amateur packet radio enthusiasts.

1. History of Packet Radio

The data packet technology was developed and put into practical use during 1960's with the ARPANET project [1]. However, the amateur world of packet radio did not benefit from the development of this technology until during the 1970's. The first amateur packet radio operations began in Montreal, Cadana, and followed by the Vancouver Amateur Digital Communication Group (VADCG) [1].

In today's radio mateur world, for some simple reasons; practicalities and economics, the packet radio technology is increasingly becoming popular. Packet radio contains wide varieties of applications for amateur radio broadcast ranging from near real-time converse mode, remote telnet, file transfer service (FTP), simple mail transfer protocol (SMTP), or even, transferring stilled images in a near real-time environment. These applications are possible because this type of technology, through a 'free' amateur radio band, is able to provide flexibility and capabilities of conventional digital communication networking at an incredibly low-cost.

For an example, an amateur packet-radio operator can sent the information through the amateur radio network to a node at may be 2500 miles away with little investment in the equipment. The minimum components of the equipment to conduct such operations are the terminal node controller (TNC), a computer or a terminal, and a 2-meter transceiver. These equipments are already available in the market at low costs. In this experiment, the PC computers under DOS are already available, hence, the only additional equipments needed are, the TNC, the transceiver, and the software to transmit radio signals in a network environments.

Further more, there are still a few more reasons that make packet radio more and more popular; transparency, error correction, and automatic control [1]. Transparency is provided by the packet radio technology to the end users. Each user make a connection to the other station, then the data input by the user will be sent to appear at the destination node automatically. This is done by the Terminal Node Controller (TNC). The TNC receives input data through the interface from the data input terminal i.e. the computer running a software packet, or a dum terminal, then, it automatically divides the data into packets and put the overheads onto each packet (usually an AX.25 protocol overheads). The TNC, then, keys up the transmitter and send the packets through the radio wave carriers i.e. UHF/VHF, or microwave, etc.

In additions, the TNC hardware also provides automatic error detection scheme. If the information is corrupted during transmission through the medium, it will automatically send a request for retransmission to the transmitter. Hence, only correct

data are received, stored, and displayed to the operator. A typical packet radio station with various components and interfaces are shown in Figure 1 below.

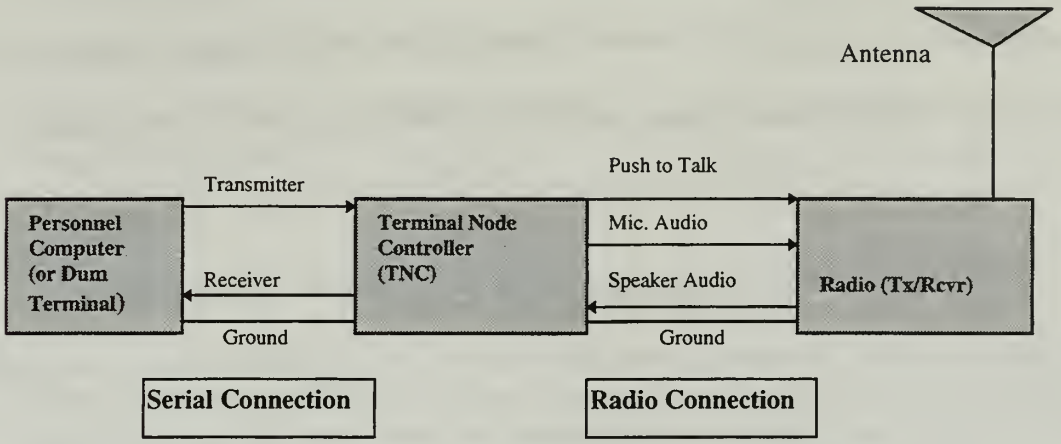


Figure 1: A Packet Radio Station [1]

One of the packet-radio features that makes the technology very interesting is that it has a shared channel capability [1]. This means that many users can share the same RF channel at the same time. This capability is possible because of the utilization of a special protocol designed for amateur packet radio called the AX.25 protocol (the Amateur X.25 protocol). The AX.25 protocol identifies the channel accessibility by using the TNC to sense if any station is transmitting, and, if it is so , it will wait until the channel is free. This scheme is called the CSMA or 'Carrier Sense Multiple Access'.

2. The AX.25 Protocol

The AX.25 protocol, or the Amateur X.25 protocol, was developed based on the conventional X.25 protocol, used widely in the wired network. The X.25 protocol is

modified due to its own lacking of some necessary features which are needed in the wireless radio types of operations. The AX.25 adds a digipeater field for extended range by other stations repeating the packets over and over again until the packets reach the destination [1]. Also, the AX.25 protocol's packet format adds the sender's callsign and the receiver's callsign to every packet, which is very useful for identification purposes [1].

The AX.25 is a link-layer protocol that is able to support various types of communication links. According to the OSI model [3], the AX.25 link-layer is just right above the physical layer, hence, it is independent of higher level and can support many communication schemes regardless of the existence of many other higher levels [2]. The independence of its link layer from higher-level layers is because it already have capabilities for reliable transfer of information across the physical link layer i.e. synchronization, error control, and flow control. These capabilities are minimum requirements for establishing a link and are providing the amateur radio operator with a lot of flexibility due to small overhead losses required by higher-layer OSI models. Typical implementation of the OSI model is as shown in Figure 2.

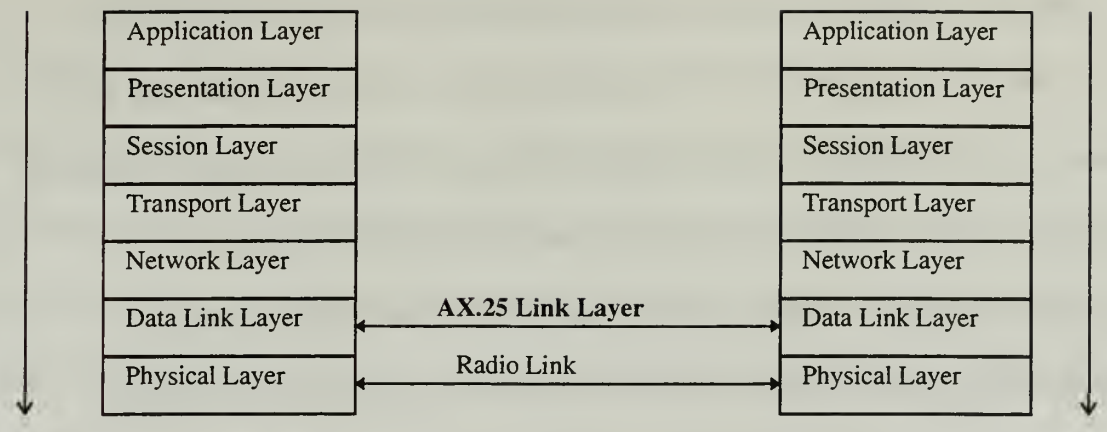


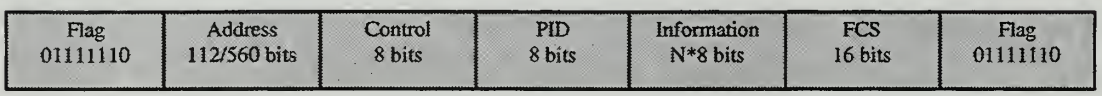
Figure 2 : The OSI Model [3]

This protocol, the AX.25, also follows the recommendations of the Consultative Committee on International Telegraphy and Telephony X.25 (CCITT X.25) [3], but it is different in the extended address field, designed to support operations in a radio environment [2]. In addition, the AX.25 also adds an Unnumbered Information frame i.e. a UI frame. The shared-RF channel is also implemented in the AX.25 protocol, following the CCITT recommendation Q.921, supported by the distinguished and extended address field [2].

The improved features of the AX.25 protocol allow the protocol to support more than one link layer per communication device, provided that the device is able to handle more than one link establishment, and also allows the protocol to be able to operate well in either 'half-duplex' or 'full-duplex' radio environments [2].

Once the data from the higher level are processed and sent to the lower level before passing it to the physical level (i.e. the link layer), the data are divided into small blocks of information called 'frames', which are composed of various smaller subsections called 'fields'. There are typically three types used by the AX.25 protocol; the Unnumbered frame (the U frame) , the Information frame (the I frame), and the Supervisory frame (the S frame). The constructions of the three types of frames are shown in Figure 3.

U-frame and S-frame structures



I-frame structure

Figure 3: The U, S, and I Frame Formats [2]

From the above frame formats, there are many field types inside the frames, and each is responding to specific function. The flag field is 8-bit or 1 octet long and has a unique specific bit sequence -- '01111110' (or, 7E Hex.). This specific bit sequence is

put at the beginning and the end of a frame to indicate that the information between the two flags is some kind of a frame. Hence, this unique bit sequence cannot appear anywhere else except in the flag fields. The mechanism that guarantee the non-reappearance of the unique flag bit sequence is called the 'bit stuffing' operations. Bit stuffing is done by monitoring the transmitted frame to see if there is any five 1's in sequence, if there is, then a '0' bit is stuffed right after the five '1' sequence. And, at the receiving DXE, it will utilize the same mechanism to monitor the frame's bit sequence, but, this time, instead of adding another '0', it will discard any '0' bit right after the five consecutive '1' sequence. This operation guarantees that there will not be any flag field pattern appearing anywhere inside the frame which may cause confusion and errors to the receiving DXE. After the beginning flag field, there is an address field which contains two sub-fields indicating both the source and destination addresses of the specific frame. In additions, if there are repeaters for the purpose of extended range along the way between the source and destination, the addresses of the repeaters are also encoded into the address field. Once, the frame reaches the destination, it may be interrupted by various sources of interference along the path, which may cause the frame errors. Hence, the AX.25 protocol provides mechanisms for error checking in the FCS field. The FCS, or the Frame Check Sum field contains a 16-bit sequence, and is calculated by both the receiver and the transmitter. The identical FCS fields on both ends indicate that there are no error, otherwise the frame is corrupted by the medium if the FCS fields are not

matched. In this case, the receiver may discard the error frame and send a request for retransmission.

The other field inside the frame which is very important is the control field. It is used to identify what type of frame it is i.e. either the U-frame, the I-frame, or the S-frame. The Information-Transfer frame, or the I-Frame, is responsible for transferring of the actual data contained in its information field [2]. To identify all the I-frame from other types, the bit # 0's in its control field is set to zero. The I-frame's control field also contains the sender's send sequence number, N(S), which is the send sequence number of the current frame. This number is updated just right before the frame is being transmitted. The sender's device has an internal send state variable to keep track of the next sequential number of the next frame which is to be transmitted. And, this internal variable is used for updating the send sequence number prior to transmitting the frame. The I-frame's control field is encoded as follows.

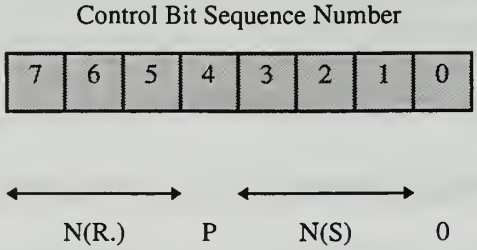


Figure 4 : The I-Frame's Control Field Format [2]

The N(R.) is the received sequence number which exists in both the I-Frame and the S-Frame. This number implies the complete and proper operations of the received frames up to the N(R.) -1 frame. Also, same as updating the N(S), the N(R.) is also updated accordingly to the device's internal received state variable, in which, the variable contains the next expected number of the incoming frame. The P-bit is actually the P/F bit (Poll or Final Operations). The Poll mode is used to request an immediate reply to a frame, and, the reply to this specific frame is done by resetting the Poll bit to a Final bit.

Other than the I-frame, the S-frame is also playing an important role in establishing and maintaining the links. The S-frame provides many valuable services such as acknowledging, requesting retransmission in case of errors , and providing link-level window control [2]. The S-frame is distinguished from others by setting bit # 0 of the control field to one, and bit # 1 to zero. In the S-frame's control field, there are a few encoding mechanisms which are really important in its operations; they are RR, RNR, and REJ. The RR is 'Receiver Ready' used to indicate that the sender of the RR is ready to receiver more I frames. It also implies that the frames up to N(R.) -1 is properly received and acknowledged. The RNR is 'Receiver Not Ready', in contrast to the RR, it is used to indicate that the sender's device is busy and not being able to receive more I frames at that time. And also, the I frames which are indexed higher than N(R.) - 1 may be lost or unacknowledged. The REJ is 'Reject Frame' which is used to indicate that the frame may be duplicated or is out of sequence, and it is rejected by the receiver. The

REJ is also, at the same time, used to request retransmission of the frames starting with the frame indexed with N(R.) and above.

Another interesting frame format used for the AX.25 protocol is the Unnumbered frame format (the U-frame). This type of frame allows more control of the link in additions to the typical Supervisory frame (S-frame). The U-frame is also responsible for establishing and terminating the link connections, and it also allows some other unusual flow operations [2]. The U-frame's control field is used to indicate either the frame is a command or response frame. There are six important encoding schemes for its control field formats, they are the SABM, DISC, DM, UA, FRMR, and the UI.

The SABM is the 'Set Asynchronous Balanced Mode' command. It is used to set both DXE's into an asynchronous balanced mode [2]. In the shared-RF channel with the AX.25 protocol, the DXE is used to replace both the master device i.e. the DCE , or the Data Communicate Equipment and the slave device i.e. the DTE, or the Data Terminal Equipment. This is because the AX.25 treats both devices as equal importance, not unequal as in the classical master/slave scenario for the typical unbalanced mode of operations. To accept the SABM request and confirm the setting of an asynchronous balanced mode, the DXE must issue the UA (Unnumbered Acknowledge) at the earliest opportunity. Once the link is established, if any DXE wants to terminate the link, it, then, has to issue the DISC command (the Disconnect Command), and wait for the UA response from the other DXE before it enters the disconnect state. When the received frame cannot be processed and the retransmission of that specific frame will not solve the

problem, the DXE will response by sending the FRMR (Frame Reject Response) to the sender containing the information field that indicates the cause of rejection. This scenario usually occurs when the received frame does not contain the FCS (Frame-Check-Sum) field, or from many other causes. In some cases, one of the DXE may try to send other frames other than the SABM or the UI while they are in the disconnected mode. Then, the DXE will response to the sender with the DM (the Disconnected Mode) response telling the other DXE that it is still in the disconnected mode and also requesting a set mode command. Another case that the DXE issues the DM response is when it receives the SABM mode and is not yet being able to establish a connection at this time.

There are some cases when the DXE wants to send frames bypassing the link layer 's normal information flow control. The UI (Unnumbered Information) frame allows the DXE to do just that. The UI frame contains the PID (Protocol Identifier) and the information field which is independent of normal flow control, and, hence, it can be passed back and forth freely [2]. But, since the UI frames are unnumbered and above the normal flow control, they will not be acknowledged by the receiving DXE, this causes unrecoverable UI frames when lost. However, the UI frame can request an indirect acknowledgment by setting its P-bit to '1'. The set P-bit causes the receiving DXE to response with a DM frame while in the disconnected mode and with either a RR or a RNR frame while in the information transferring state.

B. PACKET RADIO NETWORKING

1. Network Operating System (NOS)

Packet radio technology is a way to send information in some sort of packet format, which encapsulates data with protocols and source/destination addresses, including some information regarding the path along the way. The real benefit of packet radio technology is that the information can be send to the destination by using a networking scheme, giving the operators more flexibility, efficiency, and effectiveness in delivering the information. The software package that supports the implementations of packet radio technology in a networking scheme is called the Network Operating System (NOS). There are many Network-Operating-System (NOS) software packages available today to support the implementations of this kind of technology such as the NET/ROM, KA9Q NOS , WNOS, JNOS, and TNOS, etc. However, many of the software packages are based on the original KA9Q NOS, written by Phil Karn [6], but with some modifications and enhancements. The main attraction for the NOS is that it supports an internationally agreed standard, which gives it the ability to operate in many existing networks and interfaces such as the packet radio networks, the telephone lines, the Local Area Networks, and the TCP/IP networks (i.e. the Internet) [5]. This makes the NOS as a very powerful communication tool because information can be routed in both the conventional wired-networks and the more flexible radio-networks.

The flexibility of the NOS comes from its supports of many protocols such as the ARMPnet (the Amateur TCP/IP Packet Radio Networks), NET/ROM, and the AX.25. The packet format is as shown in Figure 5.

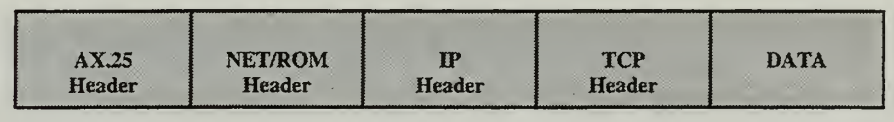


Figure 5 : NOS Multi-Protocol Frame Format [5]

From the format above, hence, there are at least three routing schemes possible by the NOS; AX.25 routing, NET/ROM routing, and TCP/IP routing. The NOS has commands to set up routing for these protocols. The 'ax25 route add' command can be used to set up a digipeater routing scheme, which will be explained later. The NET/ROM and the TCP/IP routing schemes can be accomplished by the NOS commands 'netrom route' and 'route' [5].

2. Networking Schemes

As mentioned earlier, packet radio is flexible and very useful because the information can be efficiently routed through various networks to reach the desired destinations. There are many networking schemes used in packet radio technologies such as the digipeaters, KA-nodes, NET/ROM, ROSE, TCP/IP, and the TexNet [1]. A digipeater is simply a repeater. It is only used to extend the range of the packets by re-transmitting any packet that is addressed to itself to the next digipeater, if any, or to the

destination node. The downfall of the digipeaters is that the source station has to wait for the acknowledgment from the final destination node. If there are many digipeaters along the path, it may waste precious time. One improvement is made over this scheme by allowing each digipeater to be able to acknowledge the received packet, hence, the acknowledge is executed at each link instead of the whole path. This makes the transfer of information a llittle more robust than just a simple digipeater scheme. This scheme is called the 'KA-nodes' type of operations.

a. NET/ROM

The first two schemes, mentioned earlier, are not really a real networking type of operations. The first attempt to set up a networking scheme is by using a local station connected to the user, then, the local station can connect the user to the others through that local station. And, if the destination is out of the local range, then, the NET/ROM local station will try to make a connection to another local station nearby again and then again, if necessary, until it can reach the desired destination. The local station, in this case, is called the NET/ROM station. The NET/ROM is a firmware i.e. the software installed on a chip [1], which is responsible for routing and making necessary connections. This method makes the information transfer more efficient since each user is only connected to its own local station instead of connecting to a distant destination.

b. ROSE

Another routing scheme other the NET/ROM, ROSE uses a static routing table [1]. The ROSE protocol maintains a static list of nodes that it can reach. And, if any user wants to utilize the list maintained by the ROSE switches along the way, then, he or she must specify the addresses of the ROSE switches in their digipeater fields until the packets can reach the other user maintained by a final ROSE switch. The trade-offs between the NET/ROM automatic routing scheme and the ROSE's static routing scheme is the reliability and maintenance. The ROSE protocol is more reliable because each node maintain a specific list that it can reach for sure, unlike the NET/ROM which may try to reach an reachable node. However, the NET/ROM can update its list automatically as a new node makes a contact with the NET/ROM station, without having to manually update the routing list. But, when conditions change and some nodes which are no longer reachable, the NET/ROM station will falsely still maintain those nodes as reachable ones.

c. TCP/IP

Another popular protocol is the TCP/IP protocol (Transport Control Protocol/Internet Protocol) which is supported by the various packet radio technologies such as the original KA9Q NOS (KA9Q Network Operating System), JNOS, and the TNOS (Tampa Network Operating System). The TCP/IP allows flexibility and compatibility in routing the packets through various existing networks. Also, there are many facilities already existed in the TCP/IP protocol such as FTP, Telnet, SMTP

(Simple Mail Transfer Protocol), and NNTP (Net News Transfer Protocol), etc. The implementation of TCP/IP routing scheme into the amateur packet radio world is by having the AX.25 rides on the top of the TCP/IP protocol.

When the AX.25 is integrated with the TCP/IP networks, the NOS supports such integration by using the TCP/IP links as a wormhole to route information. The command that is supported by the NOS to do the wormhole routing is the 'AXIP' command which implies 'AX.25 over IP networks'. This command can also be used with the 'attach' command to make the NOS recognizing the interface. The 'attach axip ...' command creates a RFC1226 compatible AX.25 frame encapsulator for transmission of the AX.25 frames over the internet [6], which makes the IP wormhole routing possible. The AX.25 wormhole routing over IP networking scheme is shown in Figure 6.

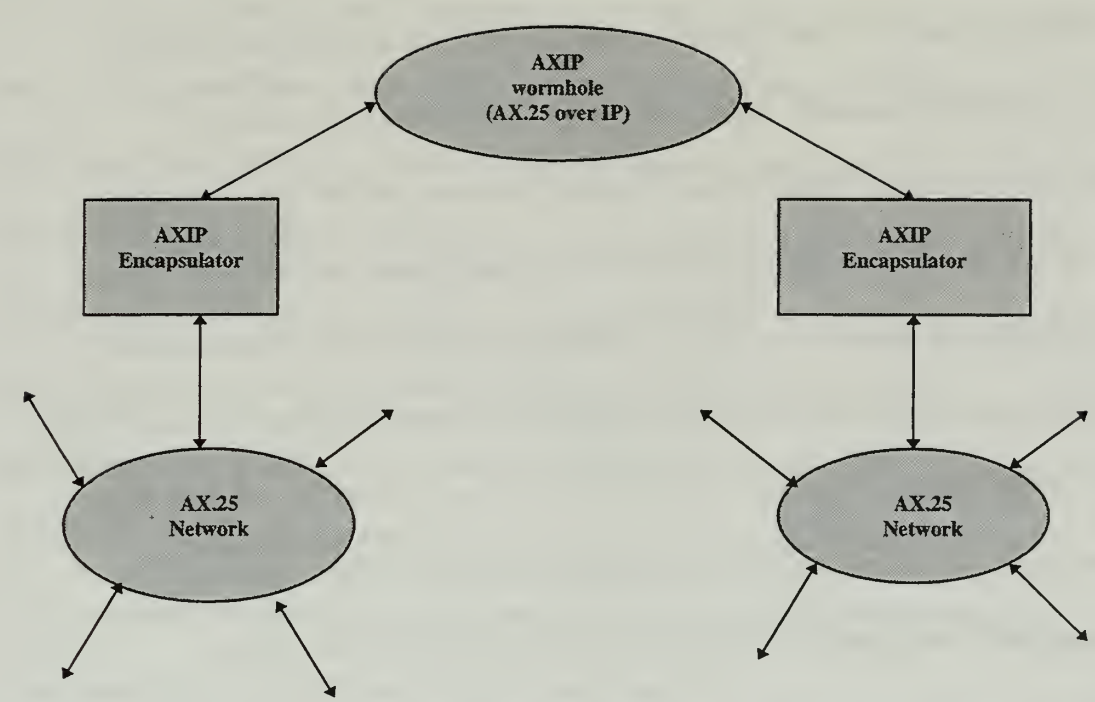


Figure 6 : AX.25 over TCP/IP Links [5]

III. EXPERIMENTAL SET-UPS

Since it is desirable to study some data-transfer characteristics and performance over different channels. A test bed with a radio channel was set up in the laboratory. Hence, several communicating nodes were set up with various communicating mediums between them. Firstly, we want to study the data-transfer behaviors over a simple radio channel. Two nodes connected with a half-duplex radio medium were used. Secondly, we want to improve the performance of the data-transfer measured from the first set-up, so, an emulated full-duplex radio channel with optimum allowable hardware configurations was used. And, thirdly, we want to study the routing effects across the node, so, we incorporated an additional node and perform data-transfer operations across the router. However, some specific hardware and software components needed to be correctly configured and interfaced are described here.

A. THE TNOS SOFTWARE

1. An Introduction to TNOS

TNOS, or the Tampa Networking Operating System, is a multi-threaded program that is able to handle the TCP/IP standard protocol in a radio environment, and, was written by Brian A. Lantz [4]. The software was developed from the original version KA9Q NOS, and, has many foundations and many features alike, but, with additional

enhancements. To make the explanations simple, we will be referring to NOS instead of the TNOS because NOS is more general and original.

NOS is a complex software package that supports most of the widely used communication protocols over the internet such as the TCP/IP, TELNET, FTP, SMTP, and over the packet radio network such as AX.25, NET/ROM, and PBBS mail [5]. The NOS can act as a gateway, a router, or simply a digipeater, and is platform-independent, which means that all these protocols can run on any operating system such as the PC, UNIX/XENIX, DEC VAX on VMS, or a Sparc workstation on SunOS, etc. [5]. Since it has an internal multitasking operating system, the software package can act simultaneously as a client, a server, and a switch for the three set of the protocols i.e. the TCP/IP, AX.25, and the NET/ROM [6]. This means that while the local user is accessing other systems through the network, other users in the network can also, simultaneously, utilize the local system's resources [6].

2. Installing the TNOS

In our experiments, the TNOS is running on a PC platform, under Windows 95 or Windows 3.11. The reason that we run the TNOS.EXE under Windows 3.11 and Windows 95 is because the software needs a DPMI (DOS Protected Mode Interface), which already exists under the Microsoft Windows 3.11 & 95 [7]. The DPMI is needed because it has capabilities to collaborate between the multitaskers and other Protected mode utilities [7]. The TNC is interfaced to the PC through a serial port i.e. the

asynchronous COM ports; COM1-COM4. The TNC is operating in a KISS mode which means that it simply passes all the information from radio to be processed by the TNOS software package inside the PC. The overall picture is as shown in Figure 7.

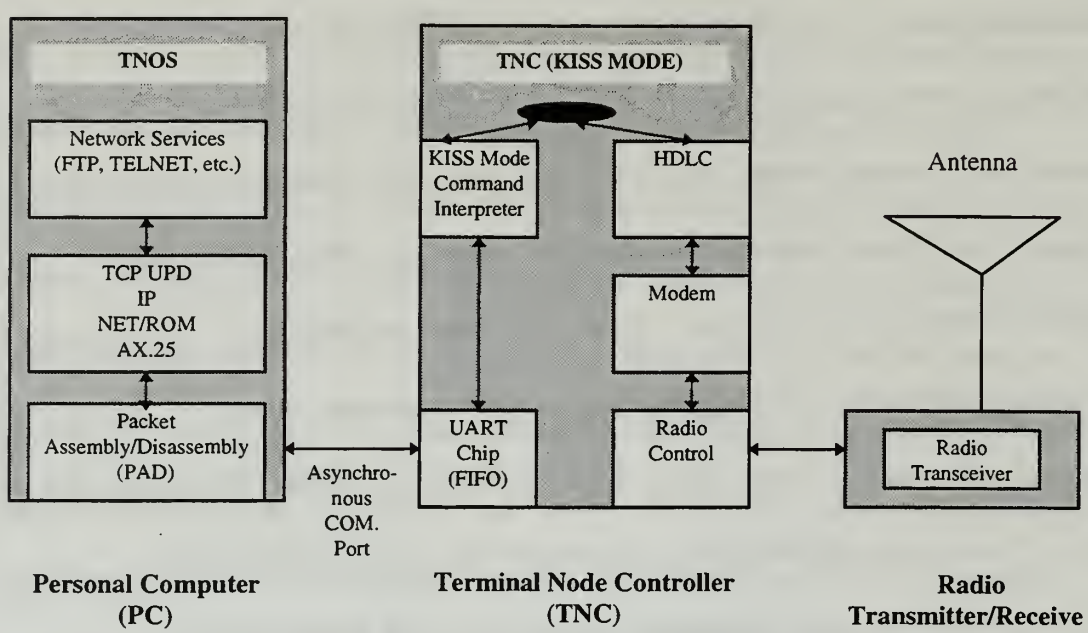


Figure 7 : The TNOS Software Interfaces [5]

The TNOS software package is distributed freely through various internet sites such as at the 'ftp://ftp.lantz.com/tnos/current' or at the 'ftp.ucsd.edu' in the '/hamradio/packet/tcpip' directory [4]. There are a couple of files needed to be downloaded; the base supported file and the execution file (TNOS.EXE). The supported file is in a '.ZIP' format, and it can be decompressed by using a free unzip shareware, i.e. the PKUNZIP.EXE with the '-d' option, which will put all the decompressed files into their corresponding directories. Once the TNOS.EXE file is downloaded into a platform i.e. a PC in our set up, then, its autoexec.nos is needed to be configured to meet specific needs for each user. When the TNOS.EXE is running, at first, it will automatically look

for the 'autoexec.nos' file which should contain all the configuration information needed for operating the network correctly. And, if the file does not exist, the user, then, has to input the detailed configurations manually line-by-line during the executing program. The example of the autoexec.nos is given in Appendix A.

3. I/O Device Interfacing

There are some important configurations needed to be carefully set up before we can operate the software correctly for our data-transfer experiments; these are the 'attach' command, the 'ftpusers' file, and the 'domain.txt' file, etc. These files are also given in Appendix A.

In our experiments we want to utilize the FTP session supported in the TNOS software to transfer the files of various sizes, and then study the file transfer characteristics through various channels and environmental settings. To accomplish this purpose, the TNOS software must be able to interface with the PC's I/O devices through their corresponding device drivers. In our case, we use the PC's asynchronous COM ports as our I/O devices. The TNOS software package can acknowledge the COM ports through its 'attach' command. The attach command allows the TNOS software to interface to various hardware device drivers such as the asynchronous COM ports, the Ethernet, and the Modem, etc. [6]. The attach command loads up the specific device driver through its corresponding device's I/O port address. Hence, the attach command will need to know the COM port's base I/O address, including its interrupt request level

(IRQ), the desired protocol to be used through that specific COM port, and some other important parameters such as the buffer size and the packet length.

4. Setting Up the FTP Session

Another important set-up before we can utilize the FTP protocol in our experiments is the 'ftpusers' file. Once the TNOS software is correctly installed, this file is under the '../etc/' directory. It maintains the allowed list of user names and their corresponding passwords, including their allowed directories and types of operations i.e. 'read files', 'create new files', or 'write/delete existing files' [5]. Also, given a domain name, and the user is trying to make a connection to another station, the software will need to know the IP address of such station. TNOS deals with this matter by having a look-up table to translate the desired domain names into their corresponding IP addresses contained in the 'domain.txt' file which can either be automatically updated, by the software itself, or, manually updated, by the user. Once the software is installed, this file is under the '../spool/' directory.

5. Setting the TNC Parameters

In our experiments, the TNC is needed to be set into the KISS mode. The TNOS has the 'comm' command that is able to set up the TNC to do such thing. Before switching the TNC into a KISS mode, to prevent unrecognized parameter-settings, the TNC should be reset into its default values before changing other parameters to other settings. The string 'reset', sent directly from TNOS to the TNC, will result in resetting

the TNC. The command that is able to send strings directly to the TNC is the 'comm' command, i.e. the communication command. To switch the TNC into the KISS mode, the strings 'int kiss', i.e. meaning 'interface kiss', must be sent to the TNC.

Other than the 'comm' command, there is another command that is able to communicate directly with the TNC and set the TNC's parameters to different settings. The TNC parameter settings can be changed by executing the 'param' command, i.e. the parameter command. The TNC's TXDELAY, PERPIST, SLOTTIME, and many other parameters can be set by using this command.

In our experiments, the TNC is operating under a KISS mode (Keep-It-Simple-Stupid Mode). The frame format at each interface is shown in Figure 8.

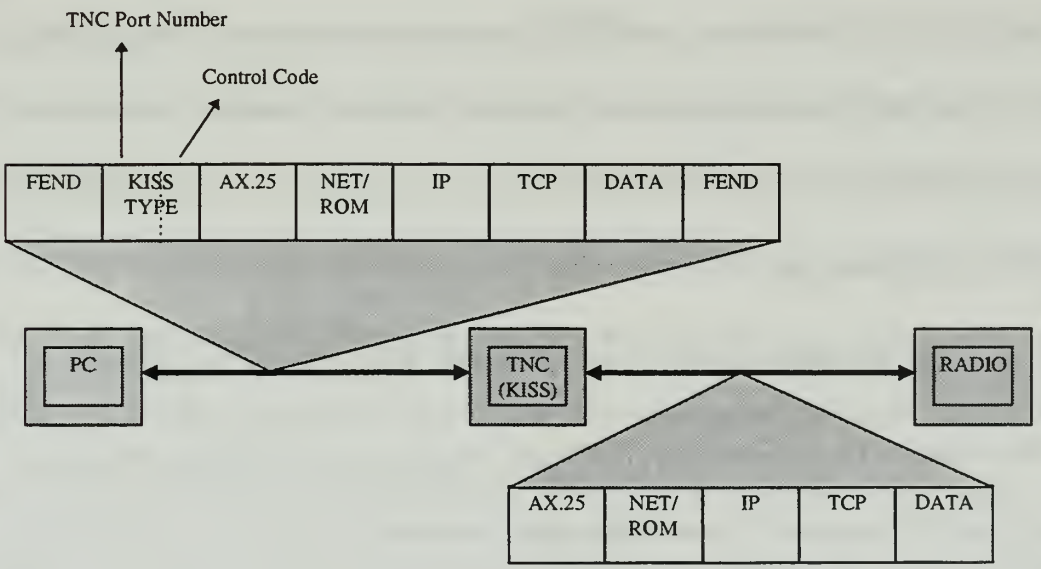


Figure 8 : The KISS Frame Format [5]

As shown by the in Figure 8, the KISS TNC simply passes all the data to the PC through the asynchronous port for the data to be processed at a higher level [5]. Hence, the TNC is only responsible for the RF channel access and frame conversion. The TNC controls the RF channel access properties by settings its two parameters; the 'PERSIST' and the 'SLOTTIME'. The RF channel is accessible when the TNC does not detect any carrier on the air. Then, it will start a timer with the length of time specified by the user through the 'SLOTTIME' parameter. Once, the timer is out, then, the TNC will generate a random number between 0 - 255. The random result will be compared with the user's specified 'PERSIST' parameter. If the generated random number is less than or equal to the user's setting PERSIST value, it will then key up the transmitter and transmitter the packets out over the channel. For frame conversion, the TNC simply puts or strips off the frame's start & finish characters i.e. the 'FEND' characters. All the decisions regarding routing, access permission, digipeating and all other higher level protocols are made at a higher level in TNOS, not at the TNC level.

B. THE HARDWARE

1. The Terminal Node Controller (TNC) & The D4-10 Transceiver

The Terminal Node Controller (TNC) and radio used in our experiments are the Katronics Data Engine and a 10- watts high-speed transceiver i.e. the D4-10 UHF Wide-Band Transceiver [19]. The transceiver operates in a high-speed packet radio mode on the UHF band (430.55 MHz). When combining the D4-10 transceiver with the

Katronics Data Engine, there are two possible speeds; the 9600 baud rate or the 19200 baud rate [19]. The Data Engine has an internal DE19K2/9K6 modem which is capable of generate the 'data carrier detected (CD)' signal from the received data stream, and is very useful for many of its operations [19].

There are quite a few fundamental mechanisms concerning radio operations which needed to be well understood before we can operate the radio channel efficiently; the 'Push-To-Talk', the 'TXDELAY', and the 'Carrier Detected (CD)'. In general, when a typical HAM radio operator wants to transmit a radio signal, the operator must push the transmitter button and then talk. This operation is called the 'Push-to-Talk' operation, and is executed whenever the operator wants to modulate the data and transmit the signal over a radio channel. However, before the signal is transmitted in the form of a radio wave through the air, the transmitter must be fully powered-up to its maximum power setting to achieve its maximum signal strength. The time delay during powering-up the transmitter is called the transmitting delay (TXDELAY). Once, the signal is transmitted and is arriving at a receiver, the receiver will have some delay-time, called the 'Squelch Time' [19], before it acknowledges the presence of the received signal and, then, produces a CD signal. This delay-time should also be accounted for in the TXDELAY setting. Once, the signal is detected, then the CD will be pulled-down low by the transceiver and output to the TNC to start demodulating the signal [19]. However, the CD may be generated by the firmware inside the TNC instead, which is a feature in all the Katronics Data Engines [19].

As mentioned earlier, the TNC in our experiments, operates in a KISS mode which allows all the higher-level protocols to be processed in the TNOS software. However, some hardware parameters inside the TNC must be optimally set up to achieve a maximum performance. These hardware parameters are the TXDELAY, PERSISTENCE, SLOTTIME, modem (half or full-duplex), and KISS [19]. Setting these parameters can largely affect the data-transfer performance of the RF channel. Generally, the smaller the TXDELAY, the better the performance due to less wait-time for the sender before keying-up its transmitter. Also, the larger the PERSIST value, the faster the data transfer because there is more chance for the sender to key up the transmitter. However, the receiver's buffers may be flooded, if the PERSIST value is set too high because the resceiver and its corresponding Central-Processing-Unit (CPU) may not be able to process the received data fast enough. Hence, there may be some lost characters due to overflowing the buffers. The lost characters will cause transmission errors and requesting for retransmission. The effect of data retransmission is negative to the data transfer performance of the channel.

Another important hardware parameter setting is the SLOTTIME. The SLOTTIME is the time period between the PERSIST algorithms to generate a random number. The SLOTTIME should be long enough to allow the packets to be processed at both ends i.e. the transmitter and the receiver, including the round-trip-estimated time on both ends. Again, intuitively, the smaller the SLOTTIME, the more frequently the transmitter makes its decisions whether it wants to transmit the packets or not (by

generating a random number and compare with the setting PERSIST value). And, further more, there is another TNC hardware parameter setting, called the 'modem', that can be set to enhance performance of the data transfer process over a radio channel by setting at full-duplex. If the FULLDUP is turned on, then the TNC will operate in a full-duplex mode. It means that, between the two communicating nodes, they will not have to wait for the channel to be free before anyone of them can transmit or receive acknowledgment data since they both have different channel of their own. Unlike operations in the half-duplex environments, with full-duplex environments both communicating nodes can transmit and receive data at the same time, which really improves the performance of the data transfer. In real life, for a full-duplex radio, there may be two frequencies needed for such operations. Also, when the operator wants the TNC to process some low-level protocols supported by its built-in firmware, not as just a dum hardware passing all the information to the TNOS at a higher level, the TNC can be commanded to exit out of a KISS mode by using the 'KISS' command.

2. The PC and its FIFO

The three PC's used in our experiments are IBM compatible with various Intel CPU's; the 66-Mhz 486DX, 200-Mhz Pentium™, and the 75-Mhz Pentium. The two PC's with the Intel 66-Mhz 486DX CPU and the 200-Mhz Pentium™ CPU are running under Windows 95. These two PC's, in our experiments, are named PC # 1 and PC # 2, consequently. The other PC with the Intel 75-Mhz Pentium CPU is running under Windows 3.11, which is named PC # 3 in our experiments.

There is a very important issue in putting various PC's together to work in a networking environment through their asynchronous I/O devices s i.e. modem and serial interfaces, which is the buffer size supported by the UART chip at their corresponding FIFO's (First-In-First-Out buffer). The UART is an abbreviation for 'Universal Asynchronous Receiver/Transmitter' [7]. The buffer mismatch may result in difficulties in transferring data through the asynchronous device due to lost data.

The performance of the UART chips depend on how fast they reset after an interrupt request for data transfer from the CPU. The original National Semiconductor 8250 chip, with a 1-byte buffer, resets in 1000 ns. [7], which is good enough in the old days where the CPU was not as fast as today. If the UART chip does not reset before the next CPU's interrupt request, there may be problems. However, with faster and faster CPU in today's market, the UART must have much faster reset time to satisfy the CPU's interrupt speed. The UART chip store data on in its buffer and wait for the CPU's interrupts to transfer data. The newer 16450 UART chip, also with 1-byte buffer, has some speed improvement – 200 ns. reset time [7]. This helps out a lot when working with faster CPU such as the 486 CPU and later. But, again, the new problems occur with increasing multitasking environments, such as those running under Windows 3.11 and Windows 95, and, with higher performance (i.e. speed) of data transfer device – 9.6 Kbaud rate and 19.2 Kbaud rate, etc. The problems originate when the 1-byte buffer is full and the UART chip is not able to transfer data to the CPU at that time because the CPU is processing other task (i.e. multitasking environment), and there is more data

coming. Hence, the UART chip may be forced to discard the data, which, in our experiments, may cause some lost characters and requests for data retransmission. The overall result is a slower speed for the FTP operation.

However, the newer UART chip, the 16550 chip, solves this problem by adding more buffers to store data to be transferred at later time. The 16550 UART chip has a 16-byte buffer, which can be set to any size between 1→16-byte size in the software (under Windows 95 in our experiments). But, the interfacing between these PC's with different buffer sizes is a problem because the 1-byte buffer may be overflowed and, hence, causes lost characters. In our experiments, PC # 1 has a 8250 UART chip, and both PC # 2 and PC # 3 have 16550AF UART chips. We solved the buffer mismatch problem by setting the 16550AF UART chip to operate with only 1-byte buffer for both receiving and transmitting. This 1-byte buffer setting works well and solves the buffer-overflowed problem in our experiments. The TNOS software provides a mean to check the asynchronous device status through the 'asystat' command. By using this command, if there is a buffer-overflowed situation, then, the number of lost characters will show up under the display 'hw over' i.e. meaning 'hardware overflow'.

3. Hardware Interfaces & Low Level Protocols

NOS (including TNOS) supports many interfacing devices such as serial ports (COM1-COM4), Modem Control, Ethernet Adapters, Clarkson Drivers, PACcom PC100, DRSI PCPA 8530 driver, High Speed DRSI/HAPN driver, Semi-port and Multi-port

KISS TNC, and etc. [5]. Also, there are a number of protocols supported by NOS to interface with such devices; for examples, the KISS protocol for TNC control, the SLIP protocol and the PPP protocol for serial-link point-to-point telephone links, the NRS protocol for NET/ROM control, and the Ethernet and ARCnet protocols for Ethernet adapters.

The SLIP (Serial-Link-Internet-Protocol) protocol can be used between a poit-to-point type of operations. The SLIP does not need a link header since the connection is only point-to-point. The IP datagrams are simply encapsulated with the SLIP frames [6]. Also, another useful protocol, used for Point-to-Point links, is the PPP protocol. The PPP encapsulates the datagrams in an HDLC-like frame, which is an Internet standard that is compatible with the CCITT (Consulative Committee in International Telegraphy and Telephony) standards [3,6]. HDLC is an abbreviation for 'High-level data link control', specified by the ISO 3309, 4335 [3]. The HDLC frame contains, consequently; an 8-bits flag field, 'at least' 1-octet address field, 8- or 16-bits control field, variable-length information field, 16- or 32-bits Frame-Check-Sequence (FCS), and again, lastly, the 8-bits flag field.

In our experiments, we use the KISS-mode TNC to interface with the PC using the AX.25 protocol to communicate with the other nodes. Also, we interface the two PC's together by using the AX.25 protocol over the asynchronous I/O devices.

IV. RESULTS

Since we want to study some data-transfer characteristics and performance over various channels, including their interactions with a radio channel, hence, there are some communication nodes needed to be set up with the desired mediums between them. Firstly, to study the data-transfer behaviors over a radio channel, we set up two nodes connecting with a half-duplex radio medium. Secondly, to improve the performance of the data-transfer from the first set-up, an emulated full-duplex radio channel with optimum allowable hardware configurations was used. And, thirdly, to study the routing effects across the node, so, we incorporated an additional node and performed data-transfer operations across the router.

A. DATA PACKETS

By using the 'trace' command, the data packets can be seen and recorded for later analysis. The 'trace' command enable the software to trace all the packets passing through the asynchronous I/O device (i.e. at the local COM port). The maximum transfer unit (MTU), a hardware dependent parameter, used for the data packets in our experiments is a 512-bytes size (specified by the attach command). Hence, each packet will have 472 bytes of data at the TCP level, because the TCP/IP cost an overhead of 40 bytes. The level hierarchy from higher-to-lower levels are TCP level → IP level → AX.25 level → KISS level, as shown in Figure 9.

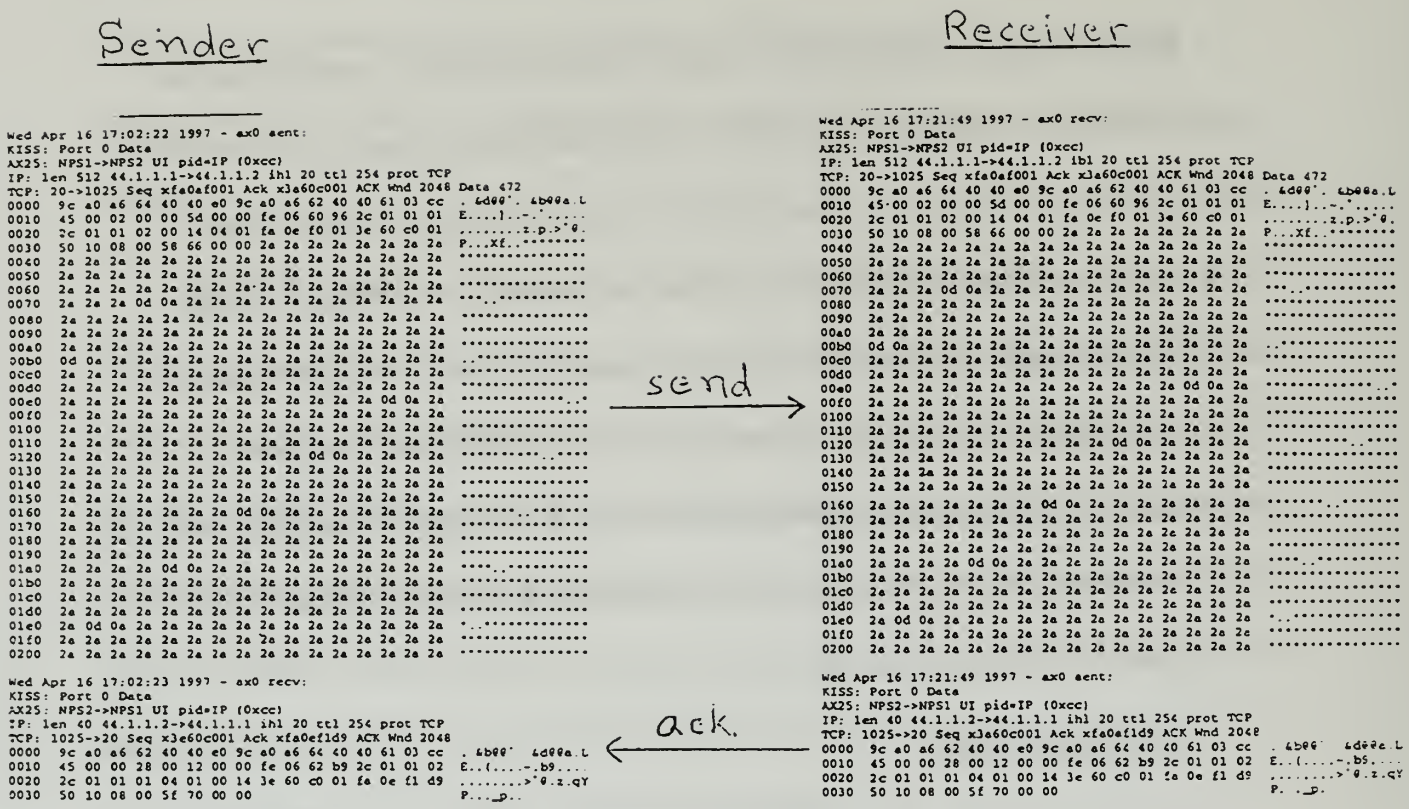


Figure 9 : Data Packet Transfer

The data-packet transfer example is a typical interaction between the sender and the receiver whenever there is a transfer of data. First, the data packet is sent by the sender through the 'ax0' port device. Then, the receiver receives the packet at its local 'ax0' port device. Then, the receiver will send an acknowledgment to the sender to allow

the sender to continue to send the next data packet. The data, in our experiments, is a series of the '*' string characters, coded as '2a'. All the actual data is displayed in 16-bytes blocks numbered starting from 0000 → 0200. At the lowest level, i.e. the KISS-protocol level between the PC and the TNC, it only maintains a port number and a control code (the format is as shown in Figure 8). In our case, as shown in Figure 9, the KISS level has a port numbered as '0'. And, the control code specifies a data type. At the next level, the AX.25 level, it maintains only a call-sign, a type of frame format, and a 'pid, i.e. a protocol identification. In our case, the frame format is a UI frame, and, the pid is an IP (or, an internet protocol), and the call-signs used for source and destination stations are NPS1 and NPS2, consequently. The IP level specifies the length of the data packet, 512-bytes long in this case. This level also specifies the IP addresses for both the source and destination. The UI frame is an 'Unnumbered Information' frame, which allows the frames to pass freely at the AX.25 level, since they are unnumbered. And, this allows the frames to be processed at higher levels i.e. the TCP/IP levels.

Since, the UI frames are processed at a higher level than the AX.25 level, the error checking is done at the TCP level in our case. If there is an error, then the TCP will report the error and automatically request a retransmission of the error packet. And, until it gets an error-free packet, it will then send an acknowledgment packet to the transmitter to allow the next sequential packet to be sent. Figure 10 is an example of a Frame-Check-Sum (FCS) error detected at the TCP level. The TCP level reports a FCS error,

and, automatically, request a retransmission. It then waits for the same packet to be re-sent and correctly received before it sends an acknowledgment packet.

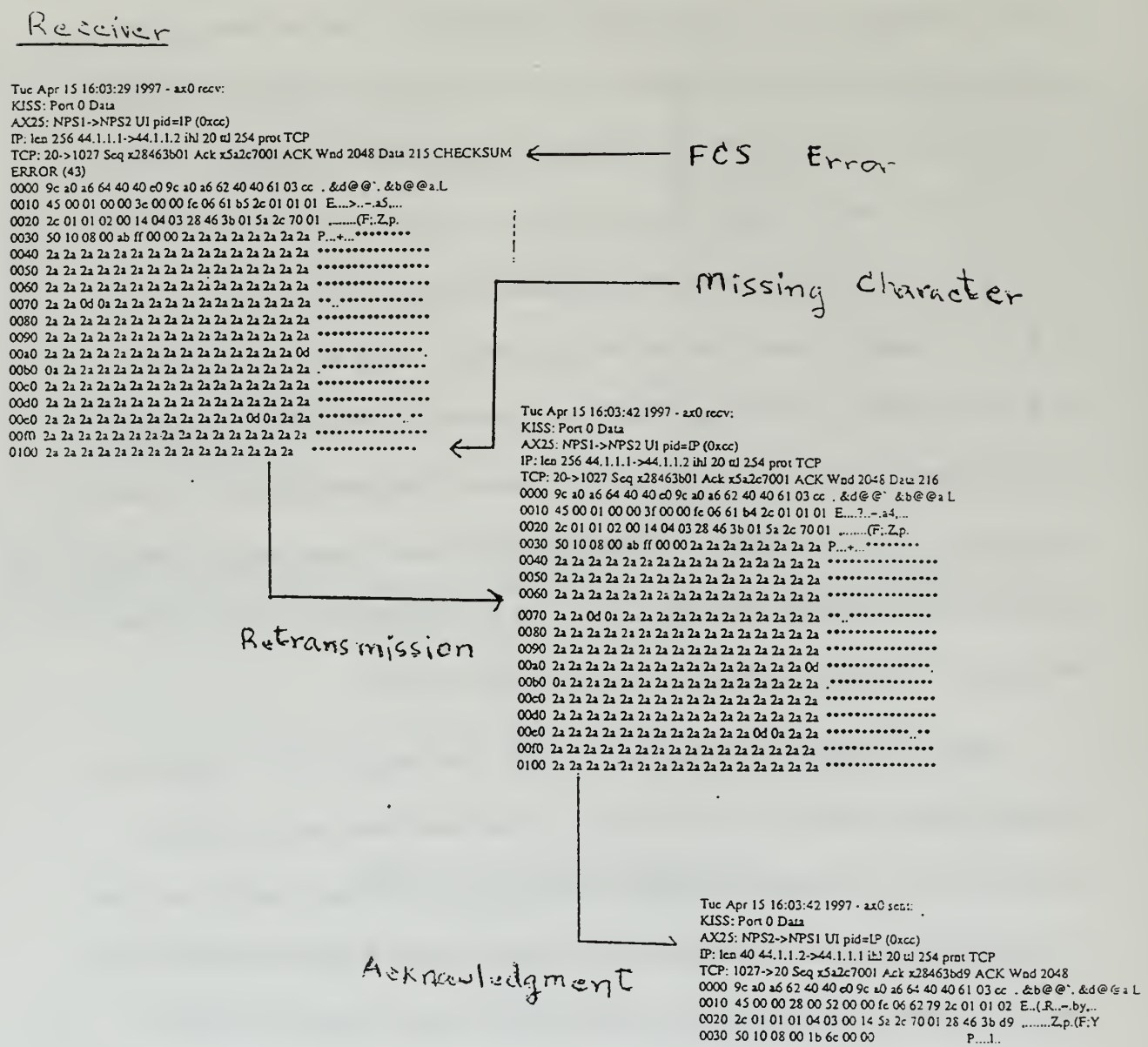


Figure 10 : An FCS Error

In this case the MTU is of 256-byte size, when accounted for a 40-bytes TCP/IP overhead, then there should be at least 216 bytes of data. But, the first data packet, as shown in Figure 10, contains only 215 bytes of the '*' characters. Hence, the TCP reports a FCS error, and automatically requests for a retransmission for the same packet. When the packet is re-transmitted and received correctly with 216 bytes of data, as shown at the TCP level of the middle packet in Figure 10, the receiver then responses with an acknowledge to notify the sender to proceed with the next sequential packet

B. EXPERIMENT # 1: HALF-DUPLEX RADIO CHANNEL

To study the data transfer behavior over a radio channel, the two nodes are set up as shown below. We decide to use a hardware flow control at the COM ports instead the software flow control because of superior performance. The hardware flow control is accomplished by selecting the 'Hardware Flow Control' option at the COM ports and setting the corresponding TNC's parameters i.e. DTR = 1 & RTS = 1 on an asynchronous (asy.) line.

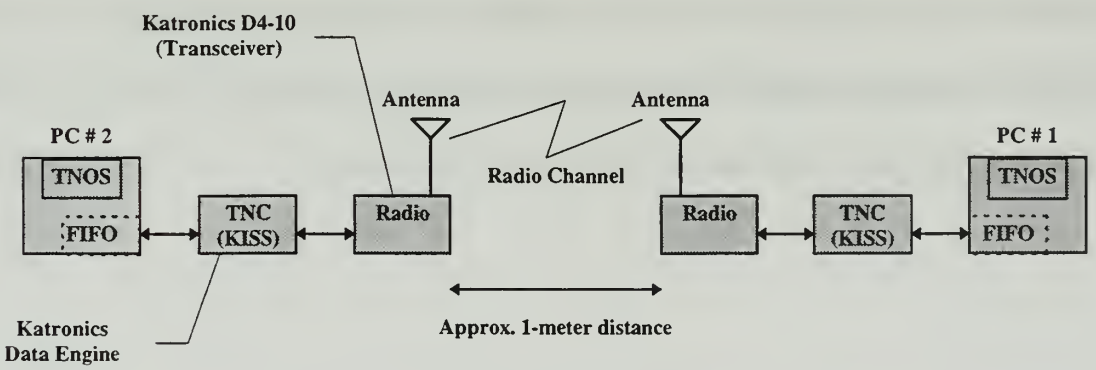


Figure 11 : A Half-Duplex Radio Channel

The TXDELAY on the TNC is set to 20 (x 10-ms unit time), i.e. 200 ms, to allow enough time for the radio to be fully powered up. Also, the PERSIST value is set to be 127, which is approximately half of a chance for transmitting. At this point, the half-duplex radio is used, and the TNC's parameter, 'modem', must be set to half duplex i.e. by using the TNC command 'modem ,half'.

The data was collected and shown in Figure 12 to 15. There are 16 files with various sizes ranging from 185 bytes, 336 bytes, 612 bytes, .., 1.884 Mbytes, consequently. Each consecutive file is roughly double the size of the previous file. There are four trials needed to be collected for each file due to the nature of a stochastic-transmitting process (which depends on the setting PERSIST value). Then, the averages in both speed and time of the four trials are derived. The speed in Bytes/second and transfer time in seconds used in the FTP operation are reported in the system at the end of each file transfer operation.

The parameter settings for Experiment # 1 are shown in Table 1. In the table, DTS is 'Data-Set-Ready'. RTS is 'Read-to-Send'. TXDELAY is 'Transmission Delay'. PERSIST is a 'Persistence' Value. SLOTTIME is a 'Slot Time'. TXTAIL is 'Transmission Tail Time'. And, lastly, FULLDUP is 'Full Duplex'.

PARAMETER	SETTING
DTS	1
RTS	1
TXDELAY	20
PERSIST	127
SLOTTIME	10
TXTAIL	0
FULLDUP	0 (off)

Table 1 : Exp. #1 Parameter Settings

Once all the parameters were set, the data samples for both the time transfer and speed versus file sizes were collected and plotted. The transfer time versus file sizes are shown in Figure 12 and Figure 13. And, the speed versus file sizes are shown in Figure 14 and Figure 15.

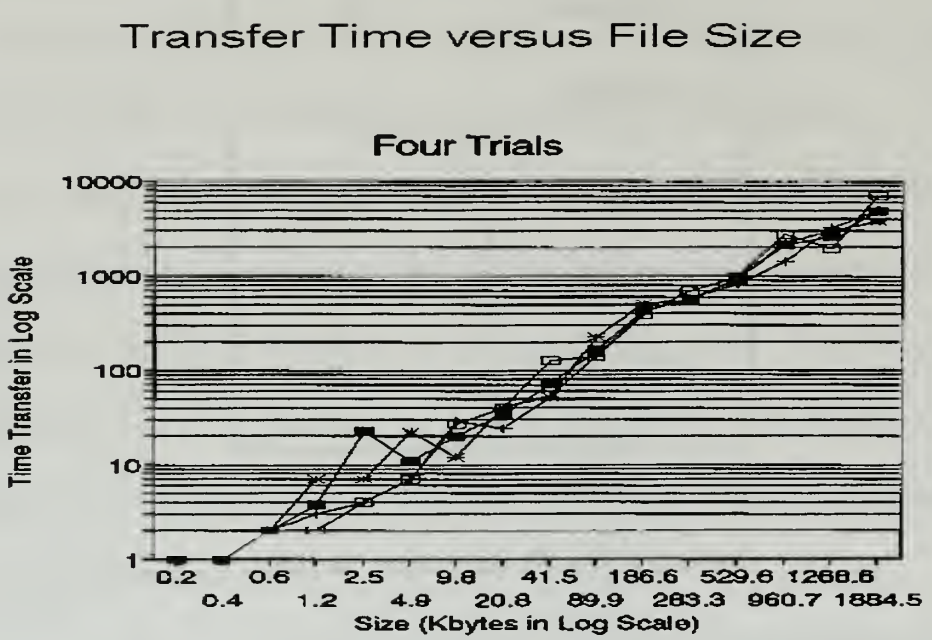


Figure 12 : Exp. # 1 Transfer Time versus File Size

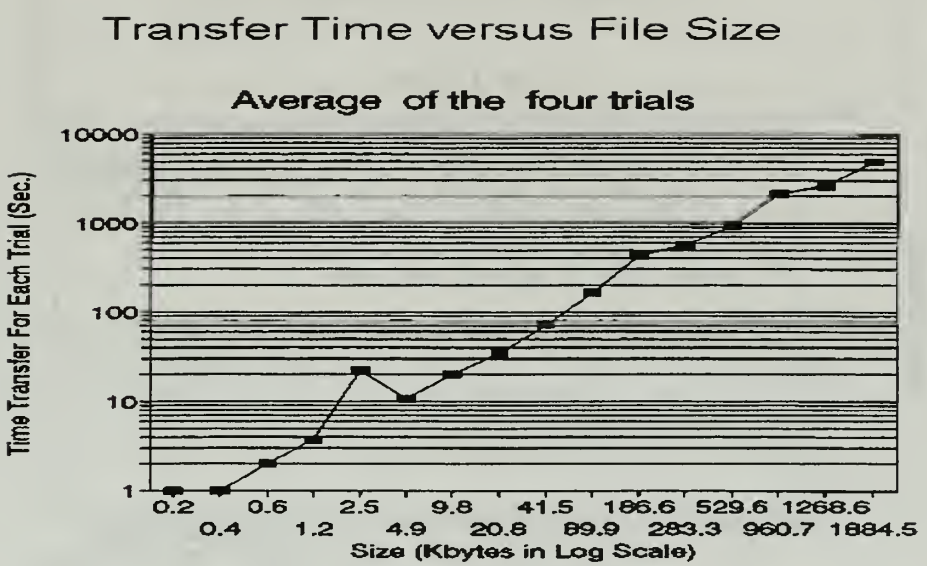


Figure 13 : Exp. # 1 Average Time versus File Size

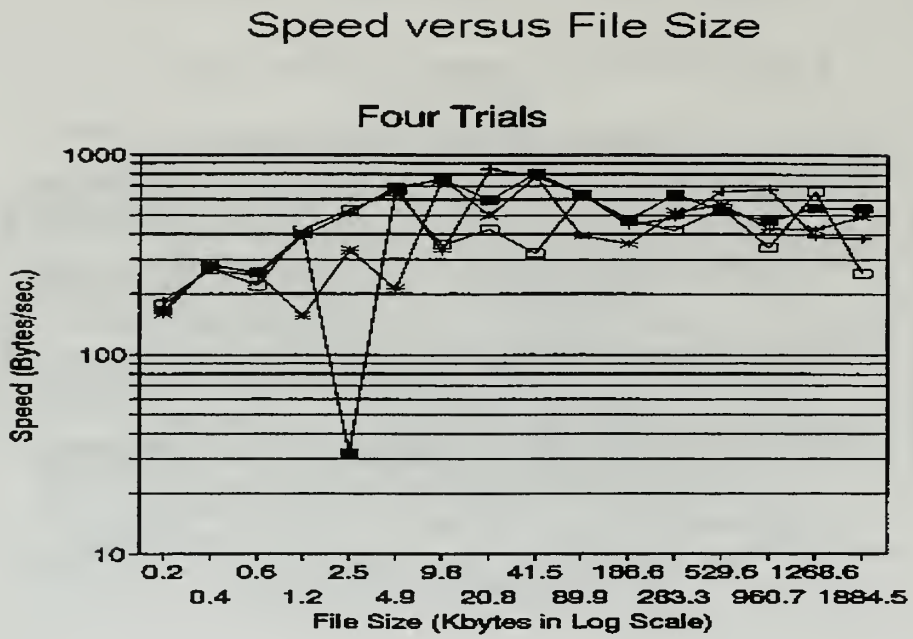


Figure 14 : Exp. # 1 Speed versus File Size

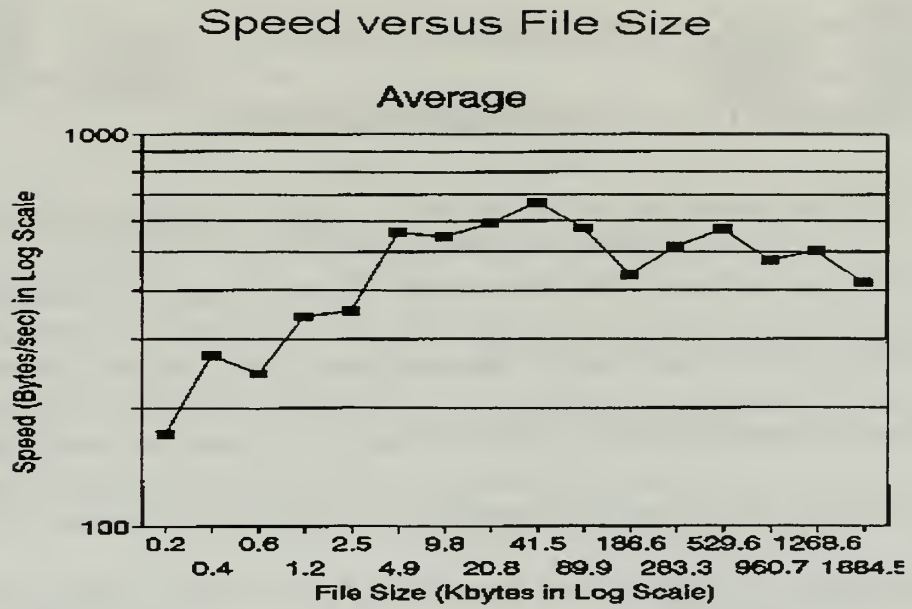


Figure 15 : Exp. # 1 Average Speed versus File Size

In the plots, both the X-axis and Y-axis are in LOG scale, and the time transfer shows linearity with respect to increasing file sizes. This is agreed with our expectation because there is a fixed-size overhead associated with each data packet, and, hence, with more data, it should should require more time to transfer the file in a linear fashion. For the average speed curve, it seems that the speed curve is saturated at some specific value after the file size is larger than 19.2 Kbytes. By taking the average of the last six data samples of the speed curve, we estimated the settled speed to be 486.70 bytes/sec.

By observations of both the transfer time and the speed curves, the data points are varied in a noticeable manner. This is because the nature of the stochastic process affected by the setting of the PERSIST value.

C. EXPERIMENT # 2: HARD-WIRED FULL DUPLEX

To improve performance of data-transfer through the radio channel, the two TNC's can be set to operate at a full-duplex mode. This will require two channels for the TNC to be able to transmit and receive at the same time. Hence, the radio must be able to operate at two different frequencies, one for receiving and the other for transmitting. However, our radio does not have such capabilities. Therefore, we emulated a full duplex environment for the two TNC's by directly connecting the two TNC's together and bypassing the radio transceivers. This emulation gives us a full-duplex like-radio channel, although we do not have a true full-duplex radio channel. The main difference between a real full-duplex radio channel and our 'emulated' full-duplex channel is the radio wave interference which may occur in the real channel. In our experiments, the interference is assumed to be minimum, since we operate in a controlled environment i.e. our laboratory. Also, fading channel is assumed to be very small, since the distance between the two nodes is very short i.e. approximately 1 meter. The set up for an emulated-radio full-duplex channel is shown in Figure 16.

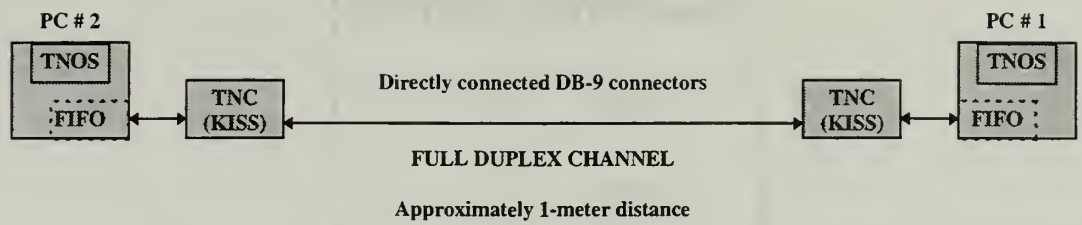


Figure 16 : A Emulated-Radio Full Duplex Channel

The connection between the two TNC's are made directly between the two 9-pins DB-15 connectors attached to the two Katronics Data Engines at port # 1's. The hard-wired diagram between the two connectors is shown in Figure 17.

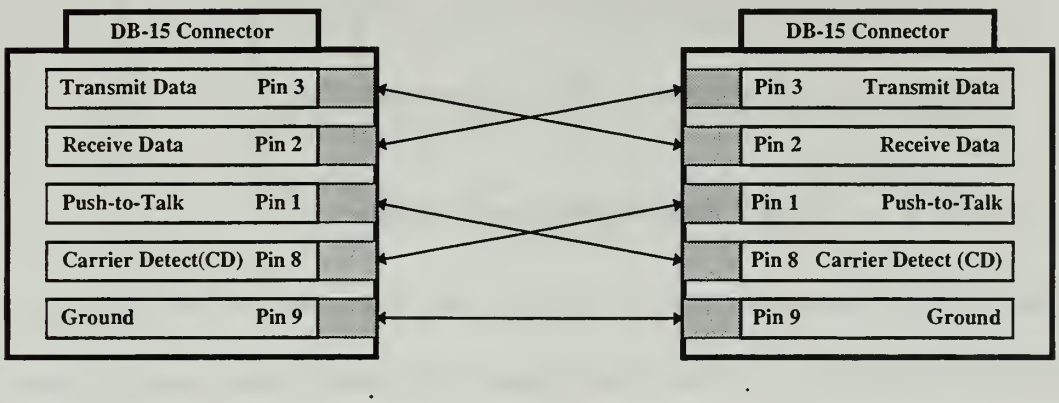


Figure 17 : TNC Hard-Wired Connections

The parameter settings for Experiment # 2 are shown in Table 2. Notice, the only difference from Experiment # 1 is the FULLDUP set to 1, which enables the full-duplex channel between the two TNC's.

PARAMETER	SETTING
DTS	1
RTS	1
TxDelay	20
PERSIST	127
SLOTTIME	10
TXTAIL	0
FULLDUP	0

Table 2 : Exp # 2 Parameter Settings

Once the all the parameters were set, the data samples for both the transfer time and speed versus file sizes were collected and plotted. The transfer time versus file sizes are shown in Figure 18 and Figure 19. And, the speed versus file sizes are shown in Figure 20 and Figure 21.

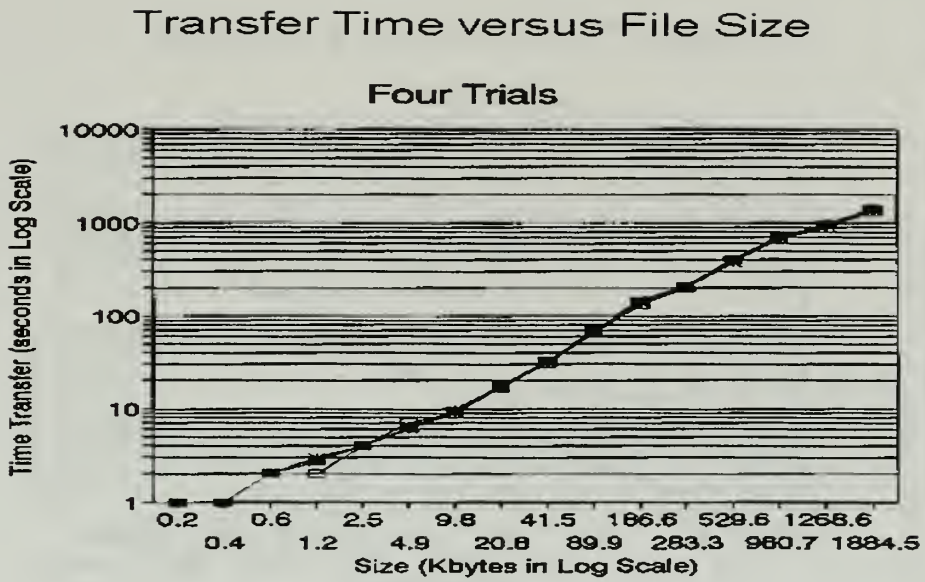


Figure 18 : Exp # 2 Transfer Time versus File Size

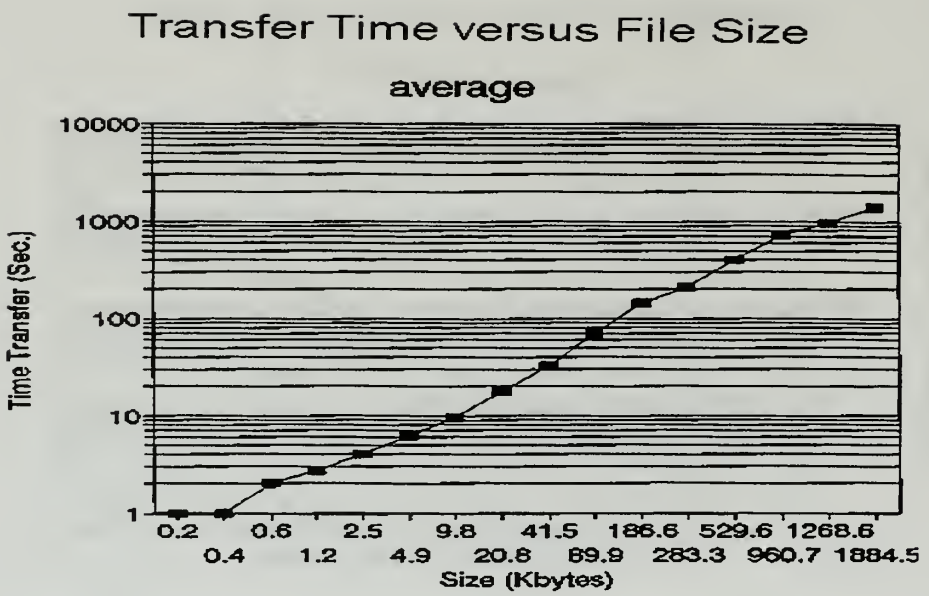


Figure 19 : Exp # 2 Average Time versus File Size

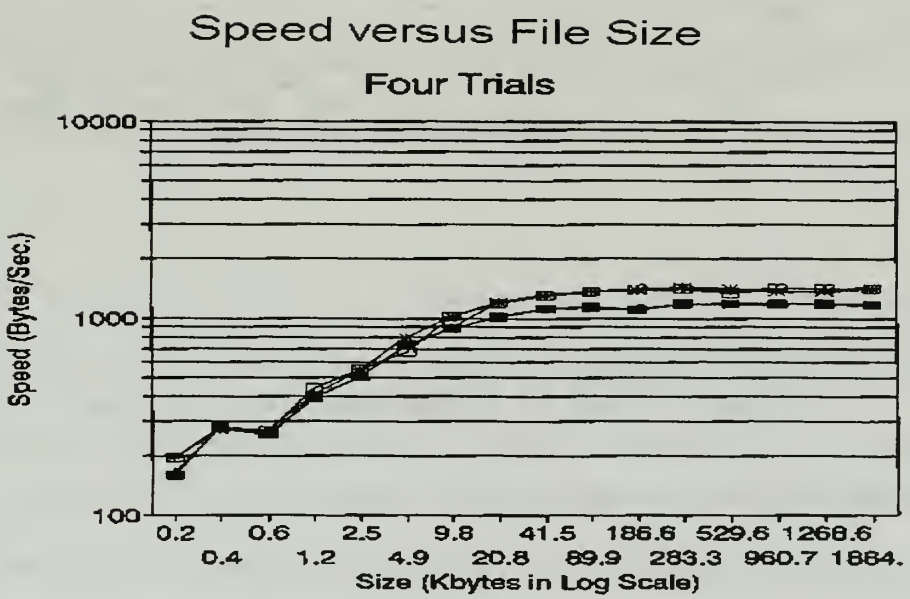


Figure 20 : Exp. # 2 Speed versus File Size

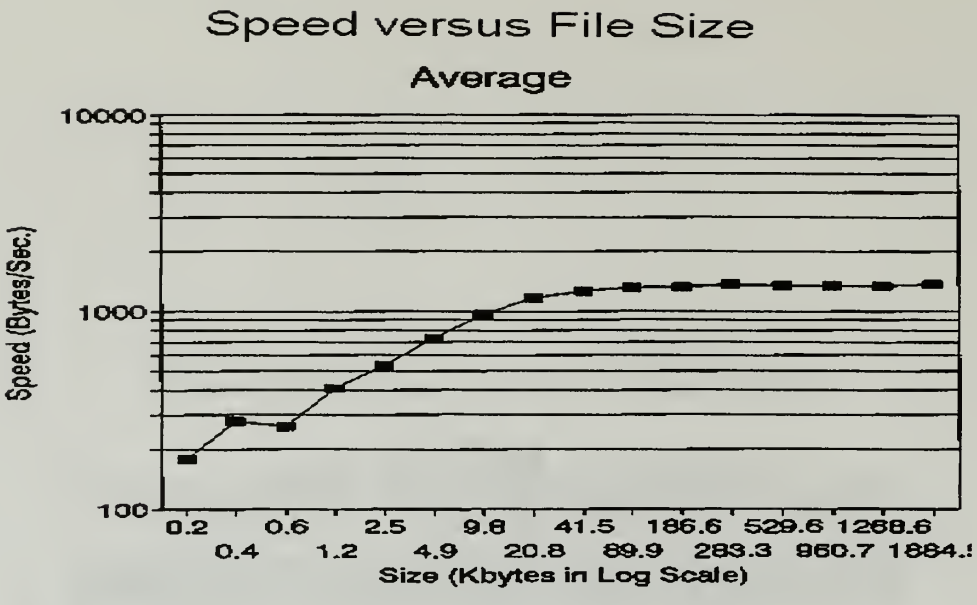


Figure 21 : Exp. # 2 Average Speed versus File Size

From the time transfer and speed curves, it seems that the FTP process is smoother than that in Experiment # 1. This is because the channel is a hard-wired full-duplex channel, in which, there is very little interference compared to the true radio channel. For the speed curve, it seems more clear than those in Experiment # 1 that the speed curve starts turning into a saturated speed at roughly 19.2-Kbytes file size. And, by taking the average of the last six data samples of the speed curve, the average saturated speed is 1344.9 Bytes/sec.

D. EXPERIMENT #3: ENHANCED HARD-WIRED FULL DUPLEX

In Experiment # 2, the channel is already a full-duplex channel but without a real radio transmitter. Hence, there are some parameters that can be set to achieve optimal performance; the TXDELAY and the PERSIST value. The TXDELAY can be set to 1 because there is no need to delay the data before transmitting for powering-up the transceiver, since there is no radio transmitter needed. Also, since the channel is relatively cleaner than that in Experiment # 1 due to hard-wired full-duplex environment, hence, the PERSIST value can be set to its maximum value (255) to maximize the chance for transmitting data, which increases the data rate i.e. speed. The parameter settings are shown in Table 3.

Once all the parameters were set, the data samples for both the transfer time and speed versus file sizes were collected and plotted. The transfer time versus file sizes are shown in Figure 22 and Figure 23. The speed versus file sizes are shown in Figure 24 and Figure 25.

PARAMETER	SETTING
DTS	1
RTS	1
TxDelay	1
PERSIST	255
SLOTTIME	10
TXTAIL	0
FULLDUP	1

Table 3 : Exp. # 2 Parameter Settings

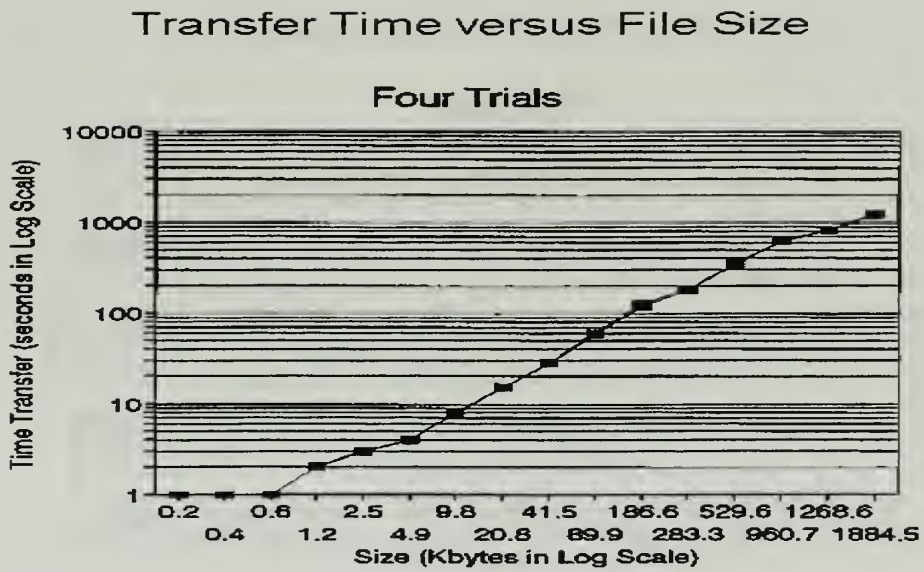


Figure 22 : Exp. # 3 Transfer Time versus File Size

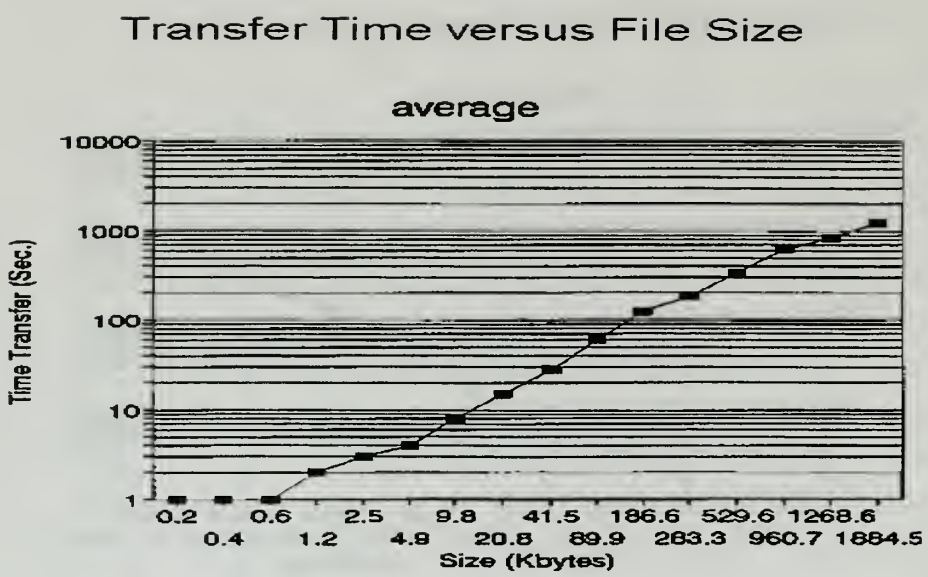


Figure 23 : Exp. # 3 Average Time versus File Size

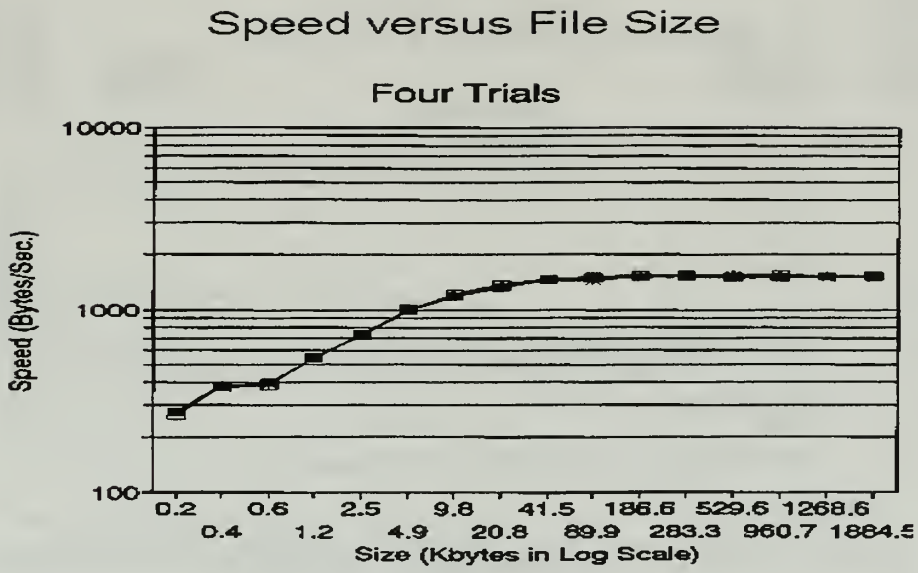


Figure 24 : Exp. # 3 Speed versus File Size

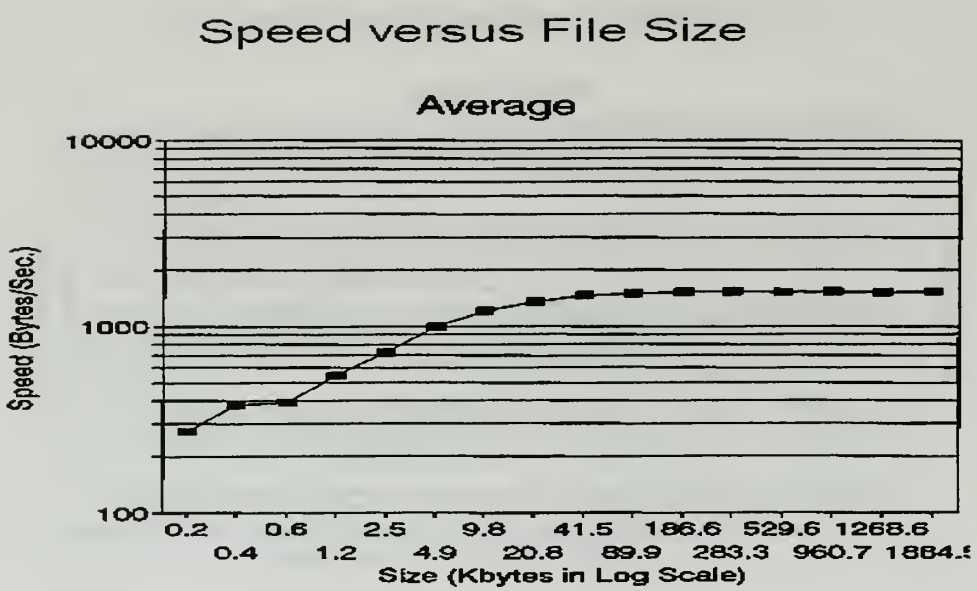


Figure 25 : Average Speed versus File Size

From the time transfer and speed curves for Experiment # 3, it is clear that there is some speed improvement when optimal parameters are used. The average speed from the last six data samples is 1528.25 bytes/sec.

E. EXPERIMENT # 4: ROUTING EFFECTS

In additions to EXP # 2, we want to study the effects of routing when there are more than two nodes involved. Hence, we make some modifications and add a router, PC # 1, between PC # 2 and PC # 3 as shown in Figure 26.

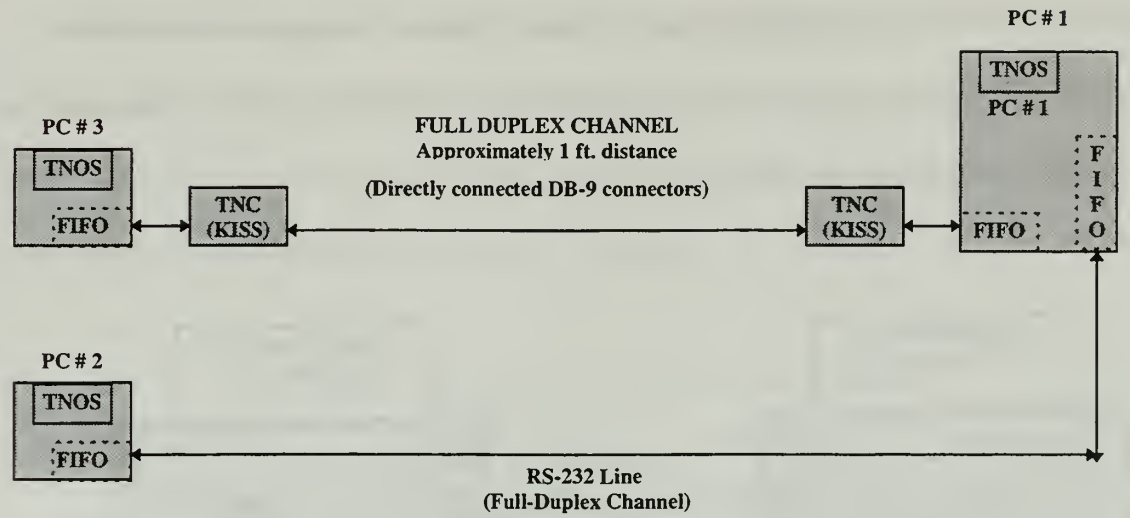


Figure 26 : Adding a Routing Node

As shown in Figure 26, PC # 1 is set up as a router between the two PC's. The FTP operations are conducted between PC # 2 and PC # 3. The files are transferred from PC # 3 to PC # 2. A route table is created and maintained at PC # 1 by using the TNOS' 'route add' command.

An additional interface between PC # 2 and PC # 1 is the RS-232 line, which was added to the set up of EXP # 2. The RS-232 line is an industry-standard physical interface. It is intended for communicating no more than 50 feet at 20,000 bps [7].

There is an important issue in dealing with the RS-232 line, it is the selection of the flow-control scheme. There are two flow-control schemes used to prevent buffer overflows; the software flow control (XON/XOFF and ENQ/ACK) , and the hardware flow control (DSR, CTS, and CD) [7]. The software flow control is executed by sending a 'STOP' character back and forth to notify the device that the buffer is already full. In contrast, the hardware flow control simply just deactivate the line to prevent/stop buffer overflows. The RS-232 link in our experiments is implemented by using two 25-pin connectors. The connection is as shown in Figure 27.

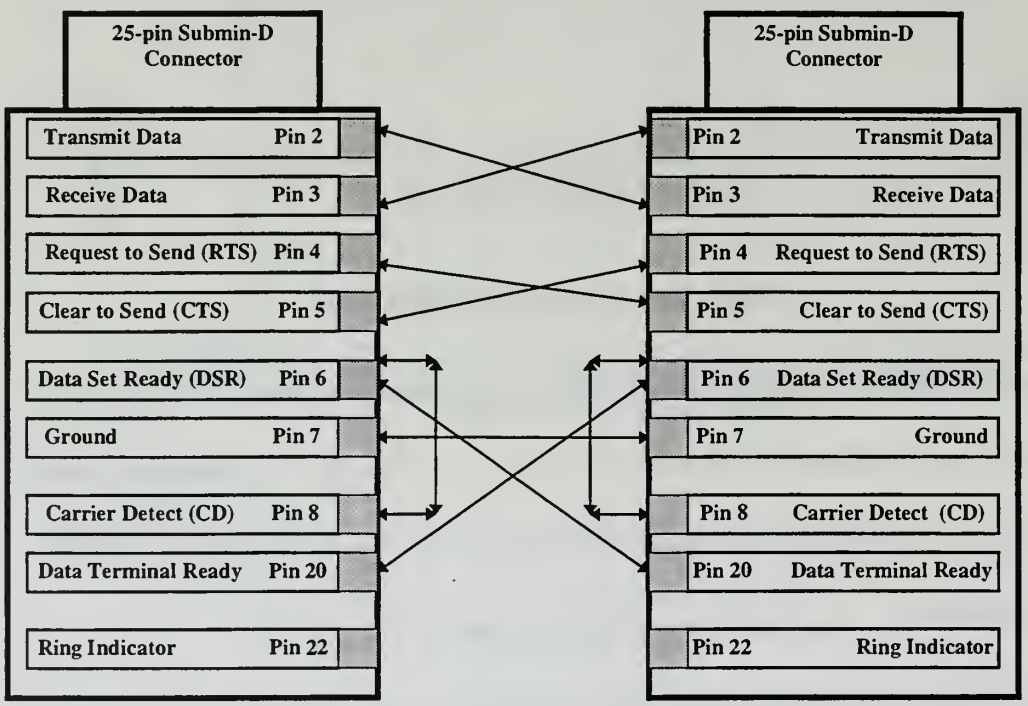


Figure 27 : RS-232 Line Connection

As seen in Figure 27, there are two lines connecting the two end nodes. Hence, the RS-232 interface in the above configuration is able to transmit and receive data simultaneously, and so, acting like a full-duplex channel. The rest of the pins, not shown in Figure 27, are not connected, because there is no need to utilize them in our experiments.

With optimal parameters setting, another router is added between PC # 2 and PC # 3 as described in the Experimental set-up. The data points were again collected and plotted. The transfer time versus file sizes are shown in Figure 28 and Figure 29. The speed versus file sizes are shown in Figure 30 and Figure 31.

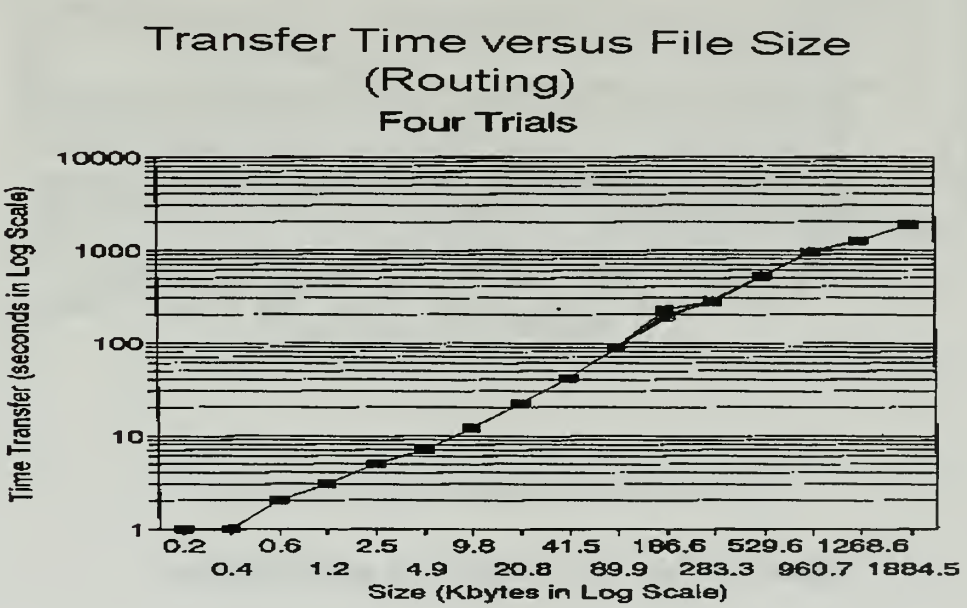


Figure 28 : Exp. # 4 Transfer Time versus File Size

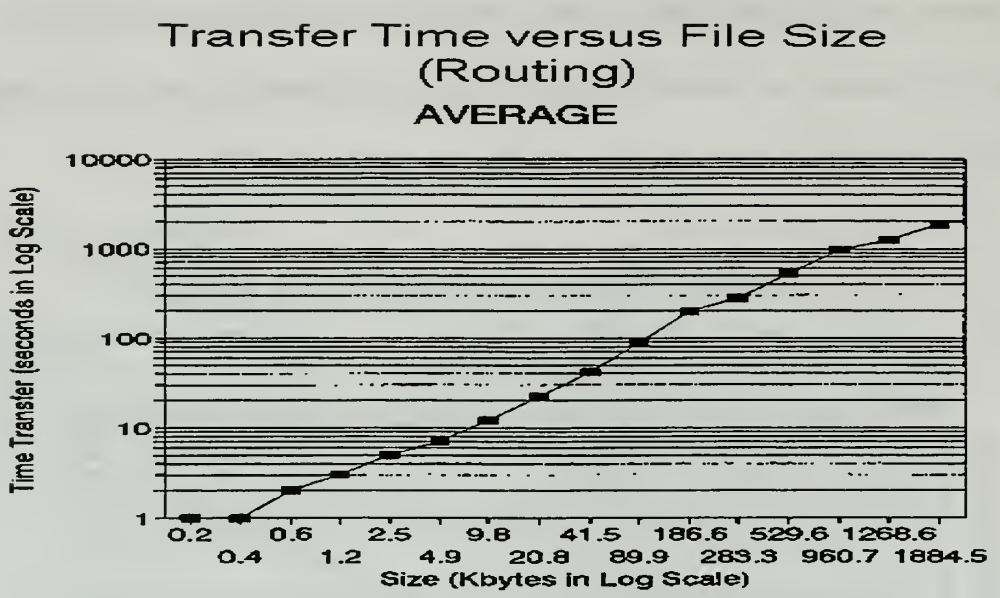


Figure 29 : Exp. # 4 Average Time versus File Size

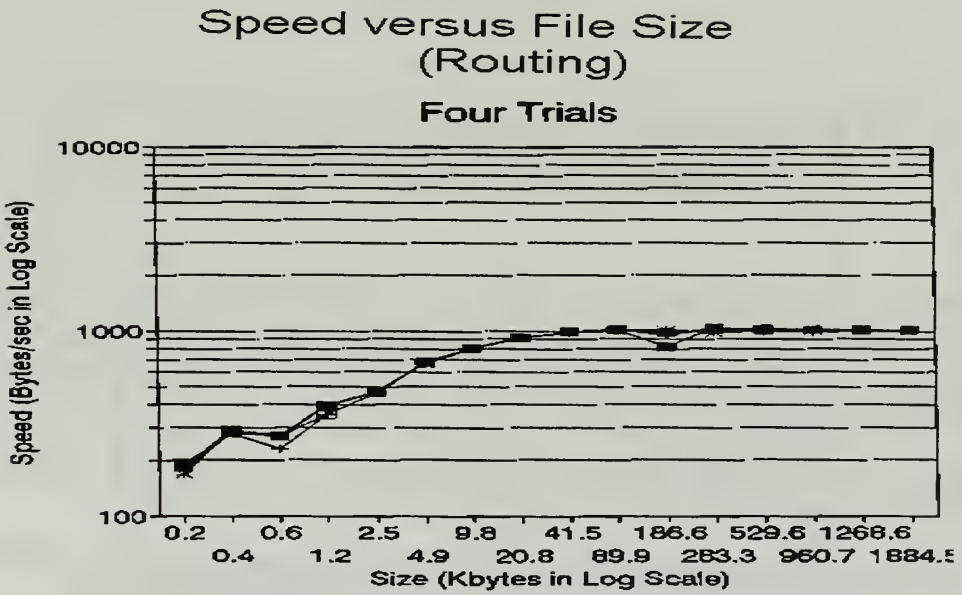


Figure 30 : Exp. # 4 Speed versus File Size

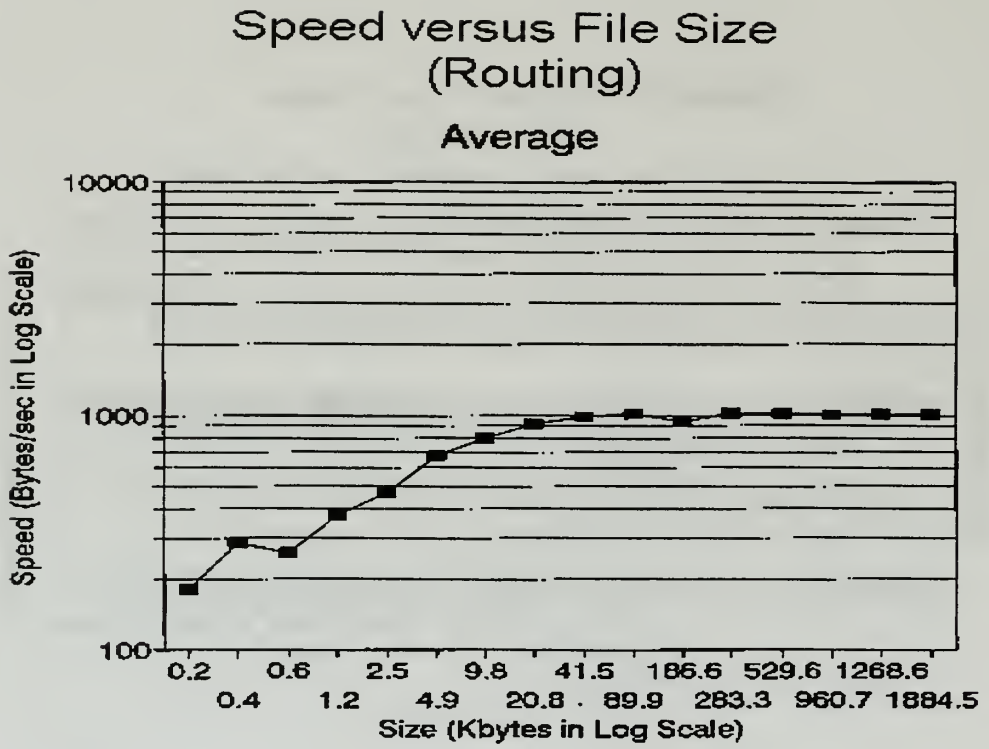


Figure 31 : Exp. # 4 Average Speed versus File Size

The average speed taken from the last six data point, in Figure 29, is 1000.2 Bytes/sec. There is a big drop in speed due to routing, compared to previous average speed curves.

V. DISCUSSION

A. TIME COMPARISON

The averages of the four curves, time transfer, are plotted and compared as shown in Figure 32.

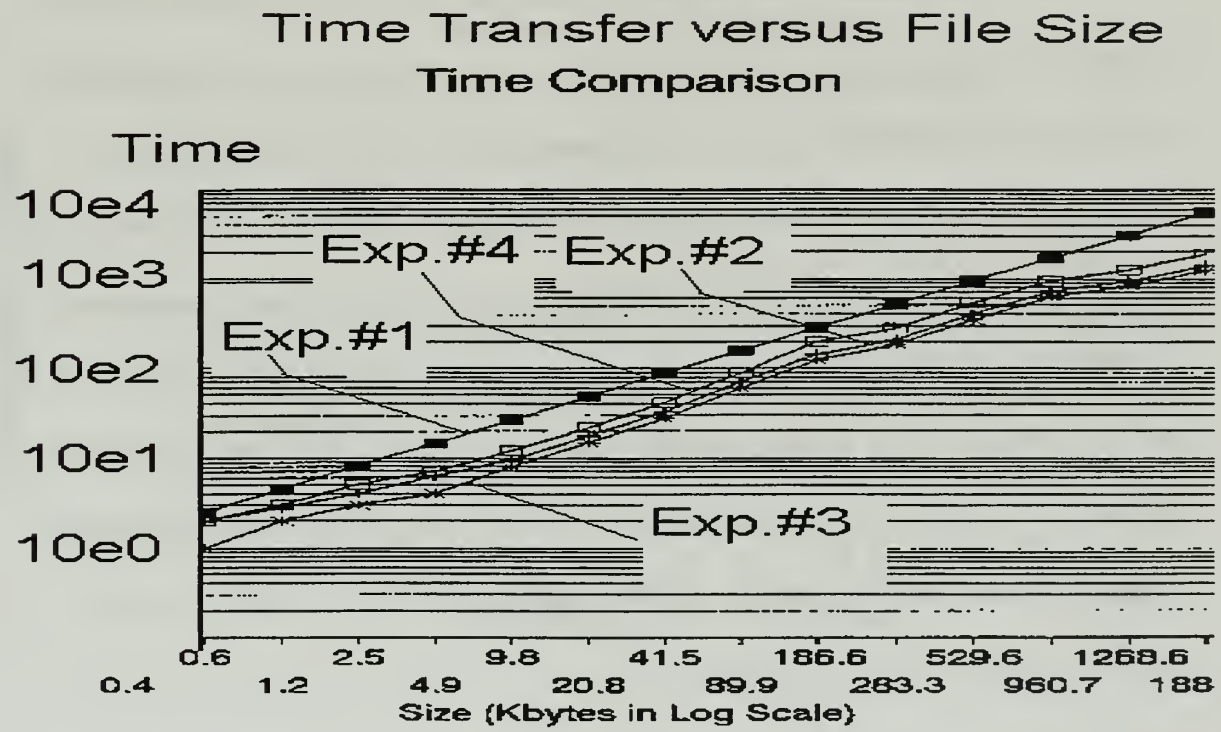


Figure 32 : Transfer Time Comparison

Figure 32 compares the time-average curves used in each experiment. From the time comparison curves as shown in Figure 32, the relative performance of each curve is related to its own position relative to others. For an example, Curve # 1 (from Exp. # 1) is at a higher position as compared to Curve # 4 (from Exp. # 4), which means that the Curve # 4 has a superior performance, since it requires less time to deliver a file of the same size.

From observations of the 4 curves, it seems that they follow the same trend even though their shapes are a little different. Curve # 1 is a straight-line shape and different from others because it is the only real radio channel. The other curves show superior performance than Curve # 1 because they are all full-duplex channels. Also, Curve # 2, Curve # 3, and Curve # 4 have similar shapes due to their similarities in the experimental set-ups. Curve # 3 has the same shape with Curve # 2 because they both have the same set-up. However, Curve # 3 has a superior performance than Curve # 2 because it has an optimal set of parameters. In addition, Curve # 4 has a very similar shape, as compared to Curve # 2 and # 3, this is because Curve # 4 has most of its the set-up like of those curves but with an additional routing delay.

From comparing Curve # 2, # 3, and # 4, it seems that at each region of the curves, they tend to follow a specific value of local slope i.e. they almost have the same slopes at some local range of file sizes. This local-slope phenomenal may be a result of the window size and/or the MTU size, used in our experiments with respect to the file size.

B. SPEED COMPARISON

For speed comparison analysis, the average speed curves of Experiment # 1 through Experiment # 4 are plotted in the same graph to demonstrate their relative performance. The plot is shown in Figure 33.

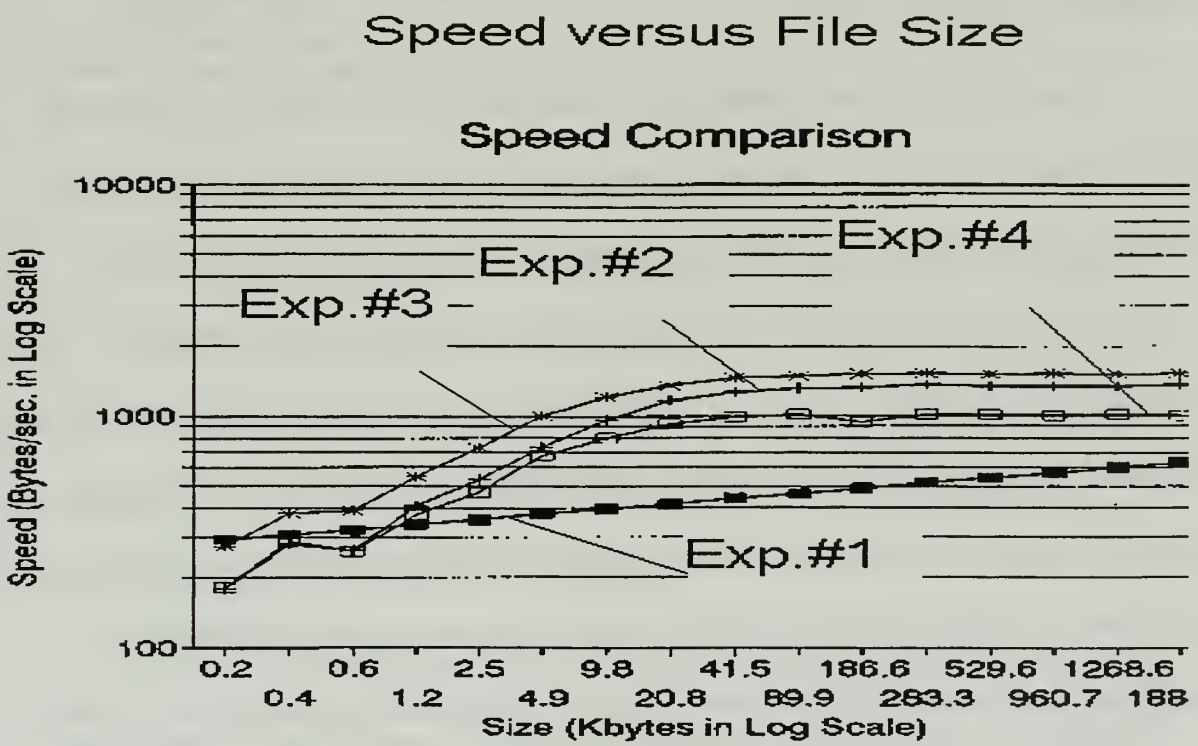


Figure 33 : Speed Comparison

Figure 33 shows the speed comparison between the four experiments. In this case, higher performance is shown in a higher vertical position of the curve. A higher-

curve position indicates a faster channel. And, as expected, like those curves in Figure 32, the performance for each curve from highest-to-lowest is Curve # 3, Curve # 2,Curve Curve # 4, and Curve # 1. The reasons are the same as those explained for Figure 32. In addition, from the observations of the curves, it is clear that, in our experiments, the full-duplex channels, Curve # 2, Curve # 3, and Curve # 4, are superior in performance than a half-duplex channel, Curve # 1.

Also, the turning corner before saturation of each curve is close to the number of 19.2-Kbytes file size. This number is related to our channel's maximum capacity of 19.2 Kbaud rate as provides by our TNC's. Before this point on the X-axis, there is an increase in speed as the file size is increasing until it reaches the channel maximum capacity of 19.2 Kbaud. The speed is increased, at the region of small file size below the channel's full capacity, because at those region the data is smaller than the pipe (i.e. an analogy for our channel capacity). Hence, the more data there is, the more the channel can deliver. However, once the data size is larger than our pipe, then no matter how large our data is we can only put certain amount of information into the pipe at one time because the pipe has a fixed size (i.e. as compared to our channel's Baud rate) . Hence, it causes a speed saturation as shown in a flat region on each curve.

In addition, from observing the curves it is clear that an additional routing node tremendously adds delay to the speed. The increase/decrease in performance, as compared to each other, is shown in Table 4.

Experiments	EXP. # 1	EXP. # 2	EXP. # 3	EXP. # 4
Speed (Bytes/sec.)	486.7	1,344.9	1,528.25	1,000.2
Speed (Bits/sec.)	3,893.6	10,759.2	12,226.0	8,001.6
Improvements (x times as Compared to Exp. # 1)	x 1	x 2.76	x 3.14	x 2.05

Table 4 : Speed Comparison

From analyzing Table 4, it is seen that the emulated-radio full-duplex channel as in Exp. # 2 and Exp. # 3 improves the speed by approximately 3 times higher than that of the radio half-duplex channel in Exp. # 1. Also, the routing node, in Exp. # 4, causes a performance drop to only 2.05 times higher than that of Exp. # 1. Hence, there is a tremendous drop in the performance due to routing.

Our maximum performance as shown in Exp. # 3 is 12.226 Kbits/sec. The rest of the bandwidth, i.e. 6.97 Kbits/sec., is probably consumed by the overheads, i.e. mainly the TCP/IP & AX.25, used in transferring the data across the mediums. Hence, the overhead cost in our experiments is relatively large compared to our available bandwidth.

In a video-conferencing application as desired in our project, the implementation may be possible only if a very effective compression scheme can tremendously reduce the size of the data stream to fit our maximum capacity. It is mentioned in the study by [9], that a high quality compression technique may be able to compress the data to below 1/50 the size of the original data. Assuming that this type of compression scheme is available and implemented with our existing system, we would be able to support a channel of more than 50 x 12.226 Kbits/sec. (i.e. 611 Kbits/sec.).

With this possible maximum capacity of 611 Kbits/sec, we may be able to support many image-delivery implementations such as delivering a quality stilled-image in a reasonable time, as studies in [9], and, may be in a distance learning application as studied in [14]. In delivering a quality tilled-image for a medium-quality image resolution of the size 230,400 bytes [9], our system would delivery in about 151 seconds, which is an acceptable time for this kind of application.

However, for a video-conferencing type of system, time of delivery is much more sensitive than just delivering a stilled image. The quality of video depends on the frame rate , i.e. number of frames per second (fps), showing on the display. A full broadcast television in North America using an NTSC standard requires about 30 fps [14]. However, this fps requires too much bandwidth. Hence, many compression standards are used to tremendously reduce the data size to fit the available bandwidth for vedio-conferencing type of application such as the H.221, H.230, H.261, and H.320, etc. [17]. Also, in the video-conferencing type of applications, an acceptable video quality may not

require as high fps as in a full quality video (i.e. it requires about 30 fps). The minimum recommendations, from the study of a distance learning type of applications [14], for the frame rate is 6 fps and for the resolution needed is 320x240-pixels. This application would require a bandwidth of 375 Kbits/sec [14]. Hence, our system with a compression ratio of approximately 4:1 would be enough to support such applications.

VI. CONCLUSION AND FUTURE WORKS

In our experiments, we implemented a half-duplex radio channel and performed a data transfer performance study. We found out that the data rate was very low and may not support video-conferencing applications. Hence, improvements were made, in experiments # 2 and # 3, by using a emulated-radio full-duplex channel. The result was an increase in performance about 3 times higher than in experiment # 1. Then, the routing effect was studied by adding a routing node between the link, the effect was a decrease in performance from experiments # 2 and # 3 down to about 2 times higher than in experiment # 1. From looking at the data rate supported by our system, it is possible that, with an advanced compression technique, the system will be able to support a video-conferencing type of applications over a full-duplex radio channel operating under a NOS.

However, before a real implementation of this type of system, there are more issues to be studied such as the security aspects of such a system, the integration of the packet-radio-networking backbone with commercial video-conferencing software/hardware packages. One way to improve security, within our experiences with the TNOS, is that we may be able to add a secure encryption/decryption scheme to the TNOS software. It is possible and practical to download the TNOS software source code from the internet site and add some security mechanisms to it before recompiling the

modified source code. Also, the integration of the video-conferencing to the backbone is possible, provided that they both support the same industrial standards such as the TCP/IP protocol, or etc.

APPENDIX A: CONFIGURATION FILES

```
# autoexec.nos file for TNOS 2.20 nps2 01/14/96
# By Narongchai Nimitbunanan


# Specify Hostname
# Local host's name.  It is used only in the greeting messages
# of the various network sservers.
# It DOES NOT set the system's IP address.

hostname nps2.ampr.org

# Set the local AX.25 address
ax25 mycall nps2

# Set thet default local IP address
ip address 44.01.01.02

# User
ax25 user nps2

############ Set AX25 parameters #######
# AX25 protocal version 2 which uses the poll/final bits
# This protocal is used when attempts to make new connections
ax25 version 2

# Number of frames that will be allowed to remain
# unacknowledged at one time on new AX25 connections
ax25 maxframe 1

# Poll threshold-- used to control retransmission behaviors
# Default value is 128
ax25 pthresh 128

# Packet lenght -- if the I-field (Information field) is greater than this, then
# it will be fragmented at the ax25 level.
# This parameter should be less than or equal to the MTU size
# ax25 paclen 256

#--------> Double MTU size
ax25 paclen 512

# Number of times trying to establish a connection.
ax25 retries 15

# Set the INITIAL value of roundtrip time in milliseconds when
# a new connection is established.
ax25 irtt 2000
```

```
# Set the AX25 idle 'keep alive' timer in millisec.
ax25 t3 60000

# Set the AX25 link 'redundancy' timer in sec.
ax25 t4 1800

# Set timer type for retransmission and recovery
ax25 timertype linear

# Set the number of byte that can be pending on an AX25 recieve queue
# beyond which I-frame will be answered with RNR (Reciever Not Ready)
ax25 window 2048

# Set the AX25 retransmission "backoff" limit for each successive
# transmission to prevent 'congestive collapse' on a loaded channel.
ax25 blimit 5

# ?????
ax25 smartroute  off

# ?????
ax25 maxwait 9000

# Set the text and time interval in seconds between broadcasts.
# Text
# ax25 bctext "Testing Beta release 01/15/97, TNOS/DOS version 2.20"
# Time interval
# ax25 bcinterval 30

# Standard PC asychronous interface (com port)
# Using com2 address 0x2f8, intrp. request 3, ax25 protocal
# , and a kiss TNC, MTU is 256 bytes
# By defaults IP datagrams are sent in UI frame

# COM1 -- This is a modem (mod0) in NPS2
#attach asy 0x3f8 4 slip mod0 1024 256 19200

#COM2
# attach asy 0x2f8 3 ax25 ax0 1024 256 19200
#--------> COM2 with double Buffer Size (2048) & Double MTU (512)
#attach asy 0x2f8 3 ax25 ax0 1024 256 19200
attach asy 0x2f8 3 ax25 ax0 2048 512 19200

# Serial Line Internet Protocol on COM 2
#attach asy 0x2f8 3 slip ax0 2048 512 19200

# Set the interface description to the strings specified
ifconfig ax0 description "440 MHz--19200 baud, TCP/IP network, D4-10 transiever"
```

```
#----- Attach an Ethernet Interface --> en0, txqlen = 8, mtu = 1500
# attach packet FxF00 ethernet 8 1500
# ifconfig en0 ipaddress 131.120.20.166


# Add a permanent entry in to the look-up table
# address resolution protocol (arp) maps the IP address into
# the subnet(link) address to the specific IP address
arp add 44.01.01.01 ax25 nps1 ax0
arp add 44.01.01.03 ax25 nps3 ax0

# Ethernet arp
# arp add 131.120.20.165 ether nps11 en0

# Add the default entry into the routing table
route add default ax0 44.01.01.01
# route add nps2 ax0 44.01.01.02
#route add nps3 ax0 44.01.01.03
# Ethernet router
# route add nps11 en0 131.120.20.165

# Invoke a device-specific control routine.  On a KISS TNC interface,
# this sends control packets to the TNC
# Enable the hardware control
param ax0 DTR 1
param ax0 RTS 1
trace ax0 211

# Domain name server
domain dns on                # Turn on
domain ttl 7200              # Time-to-live of domain name server in sec.
domain addserver 44.01.01.01   # nps1 is also a domain name server--hostid
domain maxwait 60000          # Set time-out for the dns
domain suffix ampr.org.       # Default domain name suffix
domain translate off          # Turn off the translation from IP address
                     # dot notation into a symbolic name
domain  subnet off           # Translate subnet IP address and broadcast addresses
domain verbose off           # Flag controlling return of a full name
domain update on             # Uodate domain file
domain cache size 200         # Local memory cache size
domain cache clean on         # Set the discard of expired resource records
domain cache wait 600         # Set the interval in sec. to wait for additional
                     # activity before updating the domain.txt file

# TCP Global parameters
tcp irtt 1500
tcp window 4096
tcp mss 966
tcp timer linear
tcp syndata on
```

```
tcp maxwait 60000
tcp trace on              # Turn on trace for tcp level

#
ifconfig encap mtu 576

# Global IP parameters
ip ttl 255

# This parameter is not used for TNOS v. 2.21
# ip encapnew on

# Server
start ax25

# Converse
start convers 3600

# Finger Server
start finger

# FTP
start ftp

# telnet
start telnet

# Start all kinds of servers
#start info
#start news
#start remote
#start smtp
#start ttylink
#start tutor

# smtp parameters
#smtp timer 600
#smtp max 4
#smtp trace 0
#smtp usemx on
#smtp batch on
#smtp header on
#smtp sendlzw on
#smtp reclzw on
#smtp bidcheck on
#smtp notify on
#smtp quiet off
#smtp t4 600
```

```
# FTP parameters
ftype i
ftptdisc 600
ftpmaxclients 2


# Converse parameters
conv mycall nps2
conv hostname nps2
conv interface ax0
conv header on
conv hmaxq 8192
conv umaxq 1024
conv t4 600
conv maxwait 120
conv motd "This is the CONVERS MODE"
conv sysinfo "New TNOS server nps2.ampr.org"

# Switch the TNC into the KISS mode
#comm ax0 " "
#comm ax0 "int kiss"
#comm ax0 "reset"

# TxDelay
# param ax0 1 20
#param ax0 1 1

# Persist 0 --> 255
# param ax0 2 127
#param ax0 2 255

#param ax0 3 10
#param ax0 4 0

# Half-Duplex
# param ax0 5 0
# Full-Duplex
#param ax0 5 1

# Time measurement in sec.
isat on

# Recoding for telnet
record on
```

```
# This file is created by Nimit... as a test file
# 23 Jan 1997

# Anonymous login requires no password
guest *  /tnos21 7

# Test
# NOTICE *****
# After the first run the password 'nimit' is hashed to be
# the 'number code' below and re write back to the replace the
# given password below
# Permission 7 is read(1) + write(2) + delete(4) = 7

nps1 46a9c5c3a3a021954125a6145cc05350 /tnos21 7
nps3 f3fd23386703a8d550c81bad8ef7ebf9 /tnos21 7
```

```
# This is a domain.txt file.
# This file translates the CALLSIGN into IP addresses
# A means 'Address'

nps3.ampr.org.  IN    A     44.1.1.3
nps1.ampr.org.  IN     A       44.1.1.1
nps2.ampr.org.  IN     A       44.1.1.2
nps11.ampr.org.        IN      A       131.120.20.165
nps22.ampr.org.        IN      A       131.120.20.166
```

APPENDIX B: COLLECTED DATA POINTS

EXP. #1

File Size in various scales				Time (SEC.)						Speed 9bytes/Sec.0					
Bytes	Kbytes	log_b10	log_b2	trial#1	trial#2	trial#3	trial#4	Sum	avg.	T#1	T#2	T#3	T#4	sum	avg
185	0.2	-0.7	-2.4	1	1	1	1	4	1	165	185	159	176	685	171.25
386	0.4	-0.4	-1.4	1	1	1	1	4	1	278	266	282	267	1093	273.25
612	0.6	-0.2	-0.7	2	2	2	2	8	2	258	247	253	221	979	244.75
1222	1.2	0.1	0.3	3	3	7	2	15	3.75	401	394	154	419	1368	342
2460	2.5	0.4	1.3	75	4	7	4	90	22.5	32	515	334	532	1413	353.25
4900	4.9	0.7	2.3	7	7	22	7	43	10.75	682	692	213	654	2241	560.25
9780	9.8	1.0	3.3	12	29	12	27	80	20	755	328	753	354	2190	547.5
20759	20.8	1.3	4.4	34	24	41	40	139	34.75	595	845	501	424	2365	591.25
41518	41.5	1.6	5.4	51	53	53	128	285	71.25	803	774	778	322	2677	669.25
89869	89.9	2.0	6.5	141	142	228	141	652	163	634	632	393	636	2295	573.75
186565	186.6	2.3	7.5	391	419	516	395	1721	430.3	476	444	361	471	1752	438
283261	283.3	2.5	8.1	444	584	547	686	2261	565.3	637	484	517	421	2059	514.75
529601	529.6	2.7	9.0	1010	810	937	975	3732	933	524	653	565	542	2284	571
960696	960.7	3.0	9.9	2062	1423	2252	2801	8538	2135	465	674	426	342	1907	476.75
1268621	1268.6	3.1	10.3	2331	3252	2988	1944	10515	2629	544	390	424	652	2010	502.5
1884471	1884.5	3.3	10.9	3493	4951	3808	7108	19360	4840	539	380	494	256	1669	417.25

Exp. #2

File Size in various scales				Time (SEC.)						Speed bytes/Sec.					
Bytes	Kbytes	log_b10	log_b2	trial#1	trial#2	trial#3	trial#4	Sum	avg.	T#1	T#2	T#3	T#4	sum	avg
185	0.2	-0.7	-2.4	1	1	1	1	4	1	159	166	196	197	718	179.5
386	0.4	-0.4	-1.4	1	1	1	1	4	1	280	279	271	279	1109	277.25
612	0.6	-0.2	-0.7	2	2	2	2	8	2	258	254	272	268	1052	263
1222	1.2	0.1	0.3	3	3	3	2	11	2.75	396	394	406	441	1637	409.25
2460	2.5	0.4	1.3	4	4	4	4	16	4	504	503	550	551	2108	527
4900	4.9	0.7	2.3	6	6	6	7	25	6.25	724	728	797	670	2919	729.75
9780	9.8	1.0	3.3	10	10	9	9	38	9.5	890	893	1028	1029	3840	960
20759	20.8	1.3	4.4	20	17	17	17	71	17.75	1016	1207	1207	1192	4622	1155.5
41518	41.5	1.6	5.4	37	31	31	31	130	32.5	1116	1311	1307	1308	5042	1260.5
89869	89.9	2.0	6.5	79	65	65	65	274	68.5	1137	1367	1370	1368	5242	1310.5
186565	186.6	2.3	7.5	167	134	134	132	567	141.8	1111	1388	1391	1403	5293	1323.25
283261	283.3	2.5	8.1	238	200	200	200	838	209.5	1188	1411	1431	1409	5439	1359.75
529601	529.6	2.7	9.0	449	378	372	395	1594	398.5	1178	1399	1422	1340	5339	1334.75
960696	960.7	3.0	9.9	814	675	703	676	2868	717	1179	1422	1364	1420	5385	1346.25
1268621	1268.6	3.1	10.3	1076	894	929	891	3790	947.5	1178	1417	1364	1423	5382	1345.5
1884471	1884.5	3.3	10.9	1606	1323	1327	1322	5578	1395	1172	1424	1420	1425	5441	1360.25

EXP. #3

File Size in various scales				Time (SEC.)						Speed bytes/Sec.					
Bytes	Kbytes	log_b10	log_b2	trial#1	trial#2	trial#3	trial#4	Sum	avg.	T#1	T#2	T#3	T#4	sum	avg
185	0.2	-0.7	-2.4	1	1	1	1	4	1	274	279	276	262	1091	272.75
386	0.4	-0.4	-1.4	1	1	1	1	4	1	377	386	375	377	1515	378.75
612	0.6	-0.2	-0.7	1	1	1	1	4	1	397	393	386	383	1559	389.75
1222	1.2	0.1	0.3	2	2	2	2	8	2	541	544	550	547	2182	545.5
2460	2.5	0.4	1.3	3	3	3	3	12	3	733	722	721	724	2900	725
4900	4.9	0.7	2.3	4	4	4	4	16	4	1005	990	984	998	3977	994.25
9780	9.8	1.0	3.3	8	8	8	8	32	8	1181	1216	1187	1218	4802	1200.5
20759	20.8	1.3	4.4	15	15	15	15	60	15	1322	1374	1358	1375	5429	1357.25
41518	41.5	1.6	5.4	28	28	28	28	112	28	1463	1461	1465	1470	5859	1464.75
89869	89.9	2.0	6.5	60	59	62	59	240	60	1491	1511	1437	1498	5937	1484.25
186565	186.6	2.3	7.5	123	123	121	120	487	121.8	1507	1508	1534	1543	6092	1523
283261	283.3	2.5	8.1	186	183	183	183	735	183.8	1518	1543	1545	1545	6151	1537.75
529601	529.6	2.7	9.0	354	342	342	353	1391	347.8	1493	1546	1547	1499	6085	1521.25
960696	960.7	3.0	9.9	636	623	624	620	2503	625.8	1509	1540	1538	1548	6135	1533.75
1268621	1268.6	3.1	10.3	835	825	838	835	3333	833.3	1518	1537	1512	1518	6085	1521.25
1884471	1884.5	3.3	10.9	1240	1225	1225	1225	4915	1229	1519	1537	1537	1537	6130	1532.5

Exp. #4

ROUTING

Full-Duplex Radio Simulated Link

File Size in various scales			Time (SEC.)							Speed bytes/Sec					
Bytes	Kbytes	Index	T#1	T#2	T#3	T#4	Sum	avg	Index	T#1	T#2	T#3	T#4	sum	avg
185	0.185	1.)	1	1	1	1	4	1	1)	191	178	167	184	720	180
386	0.386	2.)	1	1	1	1	4	1	2)	280	280	292	292	1144	286
612	0.612	3.)	2	2	2	2	8	2	3)	270	230	273	270	1043	260.75
1222	1.222	4.)	3	3	3	3	12	3	4)	392	353	396	352	1493	373.25
2460	2.46	5.)	5	5	5	5	20	5	5.)	467	461	467	462	1857	464.25
4900	4.9	6.)	7	7	7	7	28	7	6.)	675	668	666	669	2678	669.5
9780	9.78	7.)	12	12	12	12	48	12	7.)	802	797	802	801	3202	800.5
20759	20.759	8.)	22	22	22	22	88	22	8)	923	924	912	922	3681	920.25
41518	41.518	9.)	42	42	42	42	168	42	9)	983	976	983	982	3924	981
89869	89.869	10.)	88	88	88	88	352	88	10.)	1016	1020	1016	1017	4069	1017.25
186565	186.565	11.)	226	196	184	191	797	199.3	11.)	822	949	1011	975	3757	939.25
283261	283.261	12.)	272	272	288	278	1110	277.5	12.)	1040	1038	980	1018	4076	1019
529601	529.601	13.)	519	529	528	517	2093	523.3	13.)	1019	1000	1002	1023	4044	1011
960696	960.696	14.)	942	965	956	943	3806	951.5	14.)	1019	994	1004	1018	4035	1008.75
1268621	1268.621	15.)	1249	1245	1250	1257	5001	1250	15.)	1015	1018	1014	1008	4055	1013.75
1884471	1884.471	16.)	1862	1875	1865	1862	7464	1866	16.)	1012	1004	1010	1012	4038	1009.5

Comparison

File Size in various scales				Comparison Time avg				Comparison Speed Avg			
Bytes	Kbytes	log_b10	log_b2	# 1	# 2	# 3	#4	# 1	# 2	# 3	#4
185	0.2	-0 7	-2 4	1	1	1	1	171 25	179 5	272.75	180
365	0.4	-0 4	-1 4	1	1	1	1	273 25	277.25	378.75	266
612	0 6	-0.2	-0 7	2	2	1	2	244 75	263	389.75	260.75
1222	1.2	0.1	0.3	3.75	2.75	2	3	342	409.25	545.5	373.25
2460	2.5	0.4	1.3	22 5	4	3	5	353.25	527	725	464.25
4900	4.9	0 7	2 3	10.75	6.25	4	7	560.25	729.75	994.25	669.5
9780	9.8	1.0	3.3	20	9.5	8	12	547.5	960	1200.5	800.5
20759	20.8	1 3	4.4	34.75	17.75	15	22	591.25	1155.5	1357.25	920.25
41518	41.5	1.6	5.4	71.25	32.5	28	42	669.25	1260.5	1464.75	981
89869	89.9	2.0	6.5	163	68.5	60	88	573.75	1310.5	1484.25	1017.25
186565	186.6	2.3	7 5	430 3	141.8	121.8	199.3	438	1323.25	1523	939.25
283261	283.3	2.5	8.1	565 3	209.5	183.8	277.5	514.75	1359.75	1537.75	1019
529601	529.6	2.7	9.0	933	398.5	347.8	523.3	571	1334.75	1521.25	1011
960696	960.7	3.0	9.9	2135	717	625.8	951.5	476.75	1346.25	1533.75	1008.75
1268621	1268.6	3.1	10 3	2629	947.5	833.3	1250	502.5	1345.5	1521.25	1013.75
1884471	1884.5	3.3	10.9	4840	1395	1229	1866	417.25	1360.25	1532.5	1009.5

LIST OF REFERENCES

1. Jones, Greg, *Introduction to Packet Radio, http://www.tapr.org/html pktfaq.html#non-AX.25*

2. Buthod, Bill, *AX.25 Packet-Radio Link-Layer Protocol : Version 2.0 October 1984,* http://www.tapr.org/tapr/html/ax25.html

3. Stallings, William, *Data and Computer Communications,* Macmillan Publishing Company, Englewood Cliffs, NJ, pp. 421-417, 1991.

4. Dent, Mike, *TNOS Frequently Asked Questions: Version 0.2 Started 27th March 1995, http://www.lantz.com/tnos/*

5. Wade, Ian, *NOSintro: TCP/IP over Packet Radio,* Dowermain Ltd., UK, pp. 1-70, 1992.

6. Karn, Phil, and Lantz, Brian A., *Tampa Network Operating System User Reference Manual,* http://www.lantz.com

7. Minasi, Mark, *The Complete PC Upgrade & Maintaince Guide 7th Edition,* SYBEX Inc., Alameda, CA, 1996.

8. Tischer, Michael, and Jennrich, Bruno, *PC Intern: The Encyclopedia of System Programming,* Abacus, USA, pp.231-246, pp. 513-552, 1980.

9. Bandy, Peter C., and Koch Daniel B., *"A LOW BANDWIDTH, STILL IMAGE TRANSMISSION SYSTEM"*, 0-708-2642-3/95/$4.00 @ 1995 IEEE, Electrical and Computer Engineering Department, University of Tennessee, 1995.

10. Ohly, Martin, and Kleinholz, Lutz, *"Multimedia Medical Conferencing: Design and Experience in the BERMED Project"*, 0-8186-5530-5/94/$3.00 @ 1994 IEEE, German Heart Institute Berlin, Augustenburger Platz 1, Berlin, Germany, 1994.

11. Lu, Yen-Wen, and Bagchi, Kallol, and Burr, James B., *"A Comparison of Different Wormhole Routing Schemes"*, 0-8186-5292-6/94/$3.00 @ 1994 IEEE, Department of Electrical Engineering, Standford University, 1994.

12. Pomalaza-Raez, Carlos, *"A DISTRIBUTED ROUTING ALGORITHM FOR MULTICAST PACKET RADIO NETWORKS WITH UNI- AND BI-DIRECTIONAL LINKS"*, 0-7803-2004/94/$4.00 @ 1994 IEEE, Department of Engineering, Purdue University, 1994.

13. Gibbon, J.F., and Little, T.D.C, "Real-Time Data Delivery for Multimedia Networks", Proc. 18th Annual Conference on Local Computer Network Sept. 1993, Minneapolis, MN, 1993.

14. Kies, Jonathan K., and Williges, Robert C., and Rosson Mary B., *Controlled Laboratory Experimentation and Field Study Evaluation of Video Conferencing for Distance Learning Applications, June 1996*, http://www.hci.ise.vt.edu /~hci/htr/HCIL-96-02/HCIL-96-02.html

15. Lin, Chunhung R., and Gerla, Mario, *Multimedia Transport in Multihop Dynamic Packet Radio Networks*, 0-8186-7216-1/95/$4.00 @ 1995 IEEE, University of California, Losangeles, CA, 1995.

16. Hudson, Rhett D., *DT-5 Enabling Technologies Desktop Video Conferencing: Introduction to Desktop Video Conferencing*, http://www.visc.vt.edu/succeed /videoconf.html#introduction

17. Hendricks, Charles E., and Steer, Jonathan P., *Videoconferencing FAQ*, http://www.bitscout.com/FAQTOC.HTM#BS1

18. Harju, Jarmo, and Kosonen, Ville-Pekka, and Li, Changhong, *"Quality and Performance of a Desktop Video Conferencing System"*, 0-8186-6680-3/94/$4.00 @ 1994 IEEE, Lappeenranta University of Technology, Lappeenrata, Finland, 1994.

19. Katronics RF Data Communications Specialists, *D4-10 UHF Wide-Band Transceiver Operator's Manual,* Katronics Inc., Lawrence, KS, 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Rd., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Ryer Rd.
   Monterey, CA, 93943-6218

3. Chairman, Code IW.....1
   Information Warfare Academic Group
   Naval Postgraduate School
   Monterey, CA, 93943

4. Professor Chin-Hwa Lee, Code EC.....2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA, 93943

5. Professor Supachai Sirayanon, Code MR.....1
   Department of Meteorology
   Naval Postgraduate School
   Monterey, CA, 93943

6. 2LT. Narongchai Nimitbunanan, Royal Thai Air Force.....6
   213/3 Srisuk Road, Soi Swangmit
   Udornthani Provice, 41000
   Thailand

3 483NPG TH 2617
10/99 22527-200

DUDLEY KNOX LIBRARY
3 2768 00366765 0